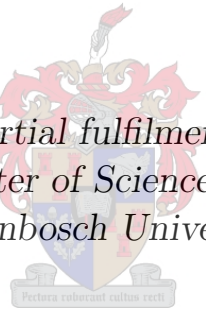
The Development of an Efficient and Secure Product Entitlement System for Pay-TV in Modern Attack Scenarios

by

Dirk Badenhorst Coetzee



Thesis presented in partial fulfilment of the requirements for the degree Master of Science in Engineering at Stellenbosch University

Supervisor: Dr. H.A. Engelbrecht
Department of Electrical and Electronic Engineering

March 2013

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2013

Abstract

A secure product entitlement system allows one party, such as a pay-TV operator, to broadcast the same collection of information to several receiving parties while only allowing a certain subset of the receiving parties to access the information. This system must still be secure in the scenario where all receiving parties who are not allowed access to the information, pool their resources in an attempt to gain access to the information. Such a product entitlement system must also be bandwidth efficient since it can be deployed in networks where bandwidth is at a premium.

The foundations of modern encryption techniques is reviewed and a survey of existing techniques, used to secure content in broadcast environments, is studied. From this collection of techniques two were identified as bandwidth efficient and are discussed in more detail before being implemented.

An attempt is then made to design a new secure bandwidth efficient encryption scheme for protecting content in a broadcast environment. Several iterations of the design is detailed, including the security flaw which makes each design insecure. The final design was implemented and compared in several metrics to the two previously selected bandwidth efficient schemes. A framework to test the correctness of the schemes over a network is also designed and implemented.

Possible future avenues of research are identified with regards to creating a secure broadcast encryption scheme and improving the software solution in which to use such a scheme.

Uittreksel

'n Veilige produk-aanspraak-stelsel stel een party, soos byvoorbeeld 'n betaal-TV-operateur, in staat om dieselfde versameling inligting na verskeie partye uit te saai, terwyl slegs 'n bepaalde deelversameling van die ontvangende partye toegelaat sal word om toegang tot die inligting te bekom. Hierdie stelsel moet steeds die inligting beskerm in die geval waar al die ontvangende partye wat toegang geweier word, hul hulpbronne saamsmee in 'n poging om toegang te verkry. So 'n produk-aanspraak-stelsel moet ook bandwydte doeltreffend benut, aangesien dit gebruik kan word in netwerke waar bandwydte baie duur is.

Die fondamente van die moderne enkripsietegnieke word hersien. 'n Opname van bestaande tegnieke wat gebruik word om inligting te beskerm in 'n uitsaai omgewing word bestudeer. Uit hierdie versameling tegnieke word twee geïdentifiseer as tegnieke wat bandwydte doeltreffend benut en word meer volledig bespreek voordat dit geïmplementeer word.

'n Poging word dan aangewend om 'n nuwe veilige bandwydte doeltreffende enkripsietegniek te ontwerp vir die beskerming van inligting wat uitgesaai word. Verskeie iterasies van die ontwerp word uiteengesit, met 'n bespreking van die sekuriteitsfout wat elke ontwerp onveilig maak. Die finale ontwerp is geïmplementeer en aan die hand van verskeie maatstawwe vergelyk met die twee bandwydte doeltreffende tegnieke, wat voorheen gekies is. 'n Raamwerk om die korrektheid van die tegnieke oor 'n netwerk te toets, is ook ontwerp en geïmplementeer.

Moontlike toekomstige rigtings van navorsing word geïdentifiseer met betrekking tot die skep van 'n veilige uitsaai enkripsietegniek en die verbetering van die sagteware-oplossing wat so 'n tegniek gebruik.

Acknowledgements

- My supervisor Dr HA Engelbrecht for his guidance;
- My family and friends for their support and motivation;
- the MIH Media lab for providing financial support for this research; and
- Irdeto for their guidance in the cryptanalysis of my implementation.

Contents

List of Figures

List of Tables

Nomenclature

Λ	Bits of security
λ	Security parameter
c	Ciphertext
k	Encryption key
m	Plaintext message
$\mathcal{A}$	Adversary
$\Pr[x]$	Probability that x will occur.
$\oplus$	The exclusive-or (XOR) operator
F_k	A keyed pseudorandom permutation with key k
$\{0,1\}^*$	A string of arbitrary length consisting of 0's and 1's.
$\{0,1\}^\lambda$	A string of length λ consisting of 0's and 1's.
$\|\|x\|\|$	Number of bits needed to represent x
$\|x\|$	The absolute value of the real number x
$x\|\|y$	The concatenation of the strings x and y
$a\|b$	The integer b is divisible by the integer a.
$a \nmid b$	The integer b is not divisible by the integer a.
$\|G\|$	The order of the group G or the size of the set G
$E(\mathbb{F}_p)$	The set of points in on the elliptic curve E with coordinates in $\mathbb{F}_p$
n	Total viewer population
$\mathbb{S}$	Set of entitled viewers

$\langle x \rangle$	A set of the values x_i for all possible values of i
$\underline{P}$	A vector P of values

Chapter 1

Introduction

1.1 Pay-TV Networks and Security

Pay-TV networks are migrating their viewers from analogue to digital transmission systems [33]. This enables them to lower the amount of bandwidth used to transmit the content to their viewers, as this is a scarce resource in terrestrial and satellite broadcast TV networks. The cornerstone of any pay-TV network is product entitlement - the ability to only allow a certain subset of their viewers to access certain broadcasts.

A typical product entitlement scenario is depicted in Figure 1.1. Alice has some information, for example a digital representation of a film, that she would like to share with some of her friends who are all called Bob. She wants to transfer her information in such a way that only the Bobs coloured blue have access to it. To complicate matters further, the colour of each Bob can change over time. A product entitlement system aims to help Alice solve this problem. An easy solution to the problem would be to only send the information to the Bobs who are allowed to access it, but in satellite and terrestrial broadcasting environments the signal can be received across a large area and this approach might not be feasible.

The field of cryptography was founded to aid parties in transferring information in such a way that only the correct receiving party can access the information. The classical cryptographic situation is shown in Figure 1.2 and has Alice wanting to send her infor-

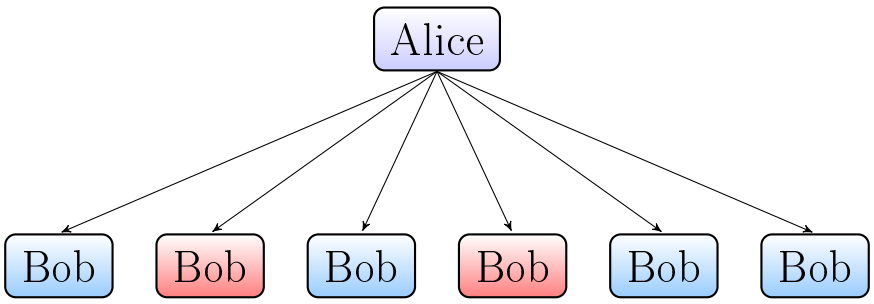


Figure 1.1: A typical product entitlement situation

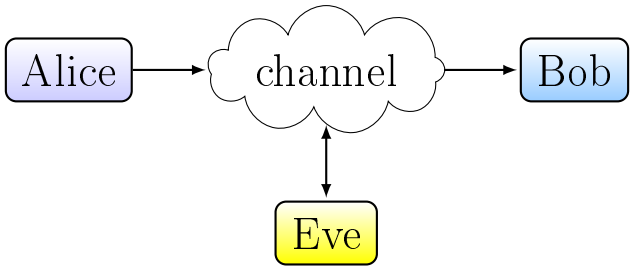


Figure 1.2: Classic cryptography situation

mation or message to only one Bob. The information is sent through some channel over which neither Alice nor Bob has complete control. Cryptography helps Alice and Bob to transfer her information in such a way that

1. Alice can be sure that only Bob can read the message, and
2. Bob can be sure that Alice is the person who sent the message.

These properties are necessary as the channel through which Alice and Bob communicate might be under the full control of a malicious third-party called Eve. It is possible for Eve to record any messages that Alice sends to Bob, perhaps stop those messages and send her own specially crafted messages to Bob in an attempt to make him reveal the information that Eve wants or to make him do something different from what Alice intended him to do.

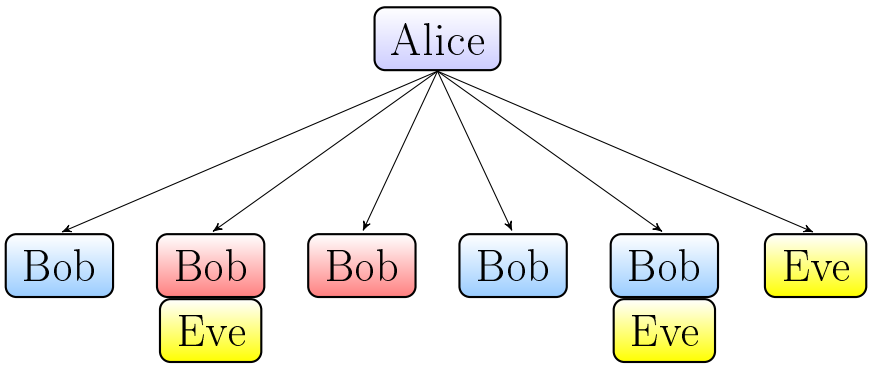


Figure 1.3: Product entitlement security concerns

It would seem that cryptography would present Alice with a solution to her problem of product entitlement. When applying cryptography to the product entitlement problem over broadcast networks the situation shown in Figure 1.2 changes to the one in Figure 1.3. From this figure we can see there are now new problems to contend with: not only is the information sent to Bob being listened to by Eve but some of the Bobs that Alice is sending her message to might actually be Eve as well. In the product entitlement situation there is no Eve that controls the channel through which Alice sends her information since each Bob-Eve can use the information they receive in any way they wish. The dynamic nature

of the subset of Bobs who are allowed to access the messages that Alice is sending means that at some stage a Bob-Eve might be in the authorised set while other times they might not, as is shown in Figure 1.3. In recent years the subsets used in product entitlement systems have become more dynamic and it is possible that the subset can change as frequently as every two hours as films are shown back to back on certain pay per view film channels [2]. Thus for Alice to ensure that only the correct subset of Bobs can access her message the product entitlement system used must be secure against this situation.

For pay-TV networks a product entitlement system would allow the pay-TV provider to present their subscribers with different types of subscriptions. One type of subscription might give a viewer access to only the sports channels while a more expensive subscription contains access to both the sports and film channels. In a pay-TV network using satellite or terrestrial broadcast networks a bandwidth efficient product entitlement system has to be used as the amount of bandwidth available to transfer entitlement messages to a viewer's device is severely constrained [21]. This system also needs to be secure so that unauthorized viewers cannot access broadcasts they are not entitled to. In recent years the attacks on pay-TV networks have escalated to such a degree that once thought secure methods, such as smart cards, are not able to prevent unauthorized viewers from accessing content [23, 26]. Thus the need exists to create secure product entitlement systems without the use of secure hardware. This is also compounded by the fact that pay-TV networks are moving into the mobile market where the use of secure hardware might not be possible [25, 28].

A field of study in cryptography that lends itself well to the product entitlement area is broadcast encryption. This kind of encryption deals with the problem of securing content in such a manner that only a certain subset of users can access the content which is set by some central authority known as the broadcasting centre. Broadcast encryption schemes are designed to be secure against collusion attacks. A collusion attack is one where several users who are not entitled to access the content, pool their information in an attempt to break the security on the content. The collusion resistance property of broadcast encryption schemes aim to solve the problem shown in Figure 1.3, but with the added constraint that all the Bob-Eves now work together.

There are several considerations that go into the design choices behind a broadcast encryption scheme. These include the bandwidth used by the scheme, the storage space required at each receiver and the time needed to compute a key. In the context of pay-TV networks bandwidth efficiency is the most important consideration as bandwidth for the widely used terrestrial and satellite broadcast networks is expensive and scarce. While several bandwidth efficient schemes have been proposed in the literature the most efficient schemes still require two keys to be broadcast [4, 12]. It is our belief that this can be brought down to one key.

The theoretical analysis of a designed broadcast encryption scheme design will give

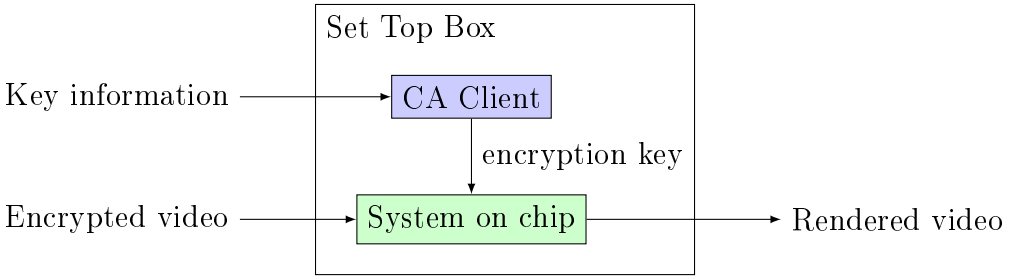


Figure 1.4: Set Top Box

insight into the bandwidth requirements, but practical measurements must be taken to validate the theoretical analysis. To this end a framework is needed that can test the broadcast encryption schemes across all the input variables for performance as well for correctness. This framework must report the measurements of all the important metrics for broadcast encryption schemes, bandwidth efficiency being the most important.

Finally, a broadcast encryption scheme on its own does not constitute a product entitlement scheme. Pay-TV networks use set-top boxes to facilitate a viewer in decrypting the content that they broadcast. A simplified view of a set-top box is shown in Figure 1.4. The key information and encrypted video is read from the information broadcasted in the network. The key information is sent to a conditional access (CA) client. The CA client can be implemented on small computational devices such as smart cards. This allows the CA client to be changed if for instance the set-top box is reassigned to a new viewer. The CA client uses the key information to calculate an encryption key that it sends to the system-on-chip. The system-on-chip decrypts and renders the video content before sending it to the display attached to the set-top box. Any designed broadcast encryption schemes will be wholly contained in the CA client. To test the proper network functionality of the scheme a framework is needed that will emulate the system on chip as well as the broadcast centre. Thus the framework has to facilitate the transfer of the secured data across the network as well as provide an interface from which the entitled set can be changed. This framework must also allow the operator to use a variety of different broadcast encryption schemes as no one scheme is perfect for every situation.

1.2 Goals

The goals of this study are thus defined as:

1. The design of a new bandwidth efficient and secure broadcast encryption scheme.

2. Evaluate the newly designed scheme and compare it to examples found in the literature.

3. Develop a framework for testing and measuring the performance of broadcast encryption schemes.

4. The development of a framework which can be used to turn a broadcast encryption scheme into a product entitlement system which tests the correctness of a broadcast encryption scheme over a network.

1.3 Contributions

This thesis details the design process of a bandwidth efficient broadcast encryption scheme as well as two supporting frameworks. Therefore the contributions made by this study are:

1. A performance testing framework for broadcast encryption schemes.

2. An evaluation of three broadcast encryption schemes using the testing framework.

3. A framework to turn a broadcast encryption scheme into a product entitlement system which can also test the correctness of broadcast encryption schemes over a network.

4. A more efficient method of calculating the values used by a scheme found in the literature, significantly improving the performance and making it a viable choice for practical use.

1.4 Overview

Chapter 2 introduces the reader to the required concepts and results in the field of cryptography to understand and design a secure broadcast encryption scheme. The chapter begins with an explanation of perfect secrecy and motivates that it is not a practical solution to implement. In order to understand the limitations of practical cryptographic implementations three ways in which security for an implementation is proved is discussed. This is accompanied by a discussion of the different kinds of abilities that an attacker might have. These abilities are divided into two categories: The attack model defines how the attacker can manipulate and extract information from the cryptographic implementation itself while the security level defines the ways in which the attacker can interact on a functional level with the implementation.

The cryptographic background discussion is split into two parts, namely symmetric and asymmetric encryption. Each of these parts is discussed individually detailing the building blocks for each and presenting some practical implementations. The chapter then focuses more on the field of broadcast encryption and discusses several schemes

found in the literature. The chapter concludes with a brief discussion on the way product entitlement systems are implemented on set top boxes used by pay-TV networks.

In Chapter 3 the design and security analysis of several broadcast encryption schemes are given. The first two schemes presented have been proposed in the literature and will serve as a benchmark against any schemes proposed in this thesis. The chapter then continues by presenting four different designs for broadcast encryption schemes as well as a cryptanalysis for each of these to show how they are insecure. Each new scheme proposed attempts to rectify the security flaw found in the previous design.

The need for a performance testing framework and a framework for turning a broadcast encryption scheme into a product entitlement system has already been discussed. Chapter 4 describes the software design and implementation of two such frameworks. The discussion starts by listing the requirements for each of these two frameworks and then gives a detailed overview of the components used for the product entitlement framework and how the components interact. These different components are then presented in more detail. Finally, the performance testing framework is presented together with a motivation for the parameters that were selected for testing as well as the procedure followed to verify that the data has been decrypted correctly. After the conceptual design of the frameworks is given, some practical issues that were encountered during the implementation of the frameworks are discussed.

Due to the different assumptions that security of the implemented schemes are based on, Chapter 5 initially discusses a way of comparing the different levels of security given by the assumptions. Next the results measured by the performance testing framework are presented and the results of our newly designed broadcast encryption scheme is compared against those of two schemes found in the literature. The chapter finally discusses the network testing of the product entitlement system and discusses the limitations encountered during the testing procedure.

Chapter 6 is a conclusion to this thesis and makes recommendations for future work as well as discussing the achievement of the goals set out for this thesis.

Chapter 2

Cryptographic Background

2.1 Introduction

This chapter aims to lay the necessary groundwork that is needed to develop a broadcast encryption scheme that is secure against collusion attacks. First an overview of the cryptographic primitives will be given. These concepts are required in order to understand the current research in broadcast encryption (covered in Section 2.6). This is followed by a detailed discussion of block and stream ciphers in Section 2.3 which is needed to understand Section 2.4 where two white-box block cipher implementations are discussed together with other similar results in related cryptographic fields. In the next section, 2.5, some number theory fundamentals and theorems are shown together with the asymmetric cryptographic schemes that can be constructed using these theorems. The chapter then proceeds to discuss some broadcast encryption schemes in Section 2.6 found in the literature and which might be candidates to implement in a product entitlement system. Finally, in Section 2.7 the functioning of a set top box is discussed as this is the hardware on which a product entitlement system will run when used in a satellite or terrestrial broadcasting environment.

2.2 Security Notions

Cryptography aims to provide information security to its users. In this section we will discuss the various definitions and models which are used in defining the security for a specific cryptographic implementation.

2.2.1 Cryptographic Schemes

Before we can proceed with the security definitions it must first be defined what constitutes a cryptographic scheme. As mentioned there are two main types of encryption schemes: symmetric and asymmetric.

A symmetric encryption scheme, also called a private key scheme, is an encryption scheme where the same key is used for the encryption and decryption of a message. For security of these schemes to hold the key must be transmitted over a secure channel.

Letting $\{0,1\}^*$ indicate a string of 1's and 0's of arbitrary length, a symmetric encryption scheme can be formally defined as [20]:

Definition 1. Symmetric Encryption Scheme

A symmetric encryption scheme is a tuple of probabilistic polynomial time algorithms $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that

1. The key-generation algorithm Gen takes as input the security parameter λ and outputs a randomized key k. It is assumed without loss of generality that the length of any key $\mathsf{k} \leftarrow \mathsf{Gen}(\lambda)$ satisfies $\|\mathsf{k}\| \geq \lambda$.

2. The encryption algorithm Enc takes as input a key k and a plaintext message $m \in \{0,1\}^*$, and outputs a ciphertext c. Since Enc may be randomized we write this as $\mathsf{c} \leftarrow \mathsf{Enc}_{\mathsf{k}}(m)$.

3. The decryption algorithm Dec takes as input a key k and a ciphertext c, and outputs a message m. We assume that Dec is deterministic, thus $m = \mathsf{Dec}_{\mathsf{k}}(\mathsf{c})$.

It is required that for every λ, every k output by $\mathsf{Gen}(\lambda)$, and every $m \in \{0,1\}^*$, it holds that $\mathsf{Dec}_{\mathsf{k}}[\mathsf{Enc}_{\mathsf{k}}(m)] = m$. This is known as the correctness property of the scheme.

An inherent problem of symmetric encryption schemes is that the key has to be exchanged over a secure channel. This is not always possible and the field of asymmetric cryptography, also known as public key cryptography, addresses this problem. Public key cryptography allows two parties, Alice and Bob, to exchange messages over an unsecured channel without compromising the security of subsequent transmissions. An asymmetric scheme has two keys - the public key, used for the encryption step, and the private key, used for the decryption step. Two parties who wish to communicate over an unsecured channel exchange their public keys over this channel. Now Alice can encrypt messages so that only Bob can decrypt them and vice versa. Formally this can be defined as [20]:

Definition 2. Asymmetric Encryption Scheme

An asymmetric encryption scheme is a tuple of probabilistic polynomial-time algorithms $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

1. The key generation algorithm Gen takes as input the security parameter λ and outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$. We refer to the first of these as the public key and the second as the private key. We assume for convenience that pk and sk each have length of at least λ and that λ can be determined from pk and sk.

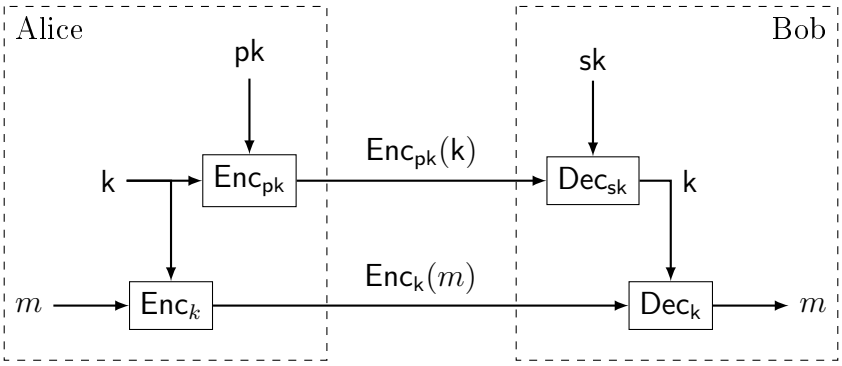


Figure 2.1: Hybrid encryption between Alice an Bob

2. The encryption algorithm Enc takes as input a public key pk and a message m from some underlying plaintext space (that may depend on pk). It outputs a ciphertext c and we write this as $\mathsf{c} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$.

3. The decryption algorithm Dec takes as input a private key sk and a ciphertext c. and outputs a message m or a special symbol $\perp$ denoting failure. We assume that without loss of generality that Dec is deterministic and write this as $m = \mathsf{Dec}_{\mathsf{sk}}(\mathsf{c})$.

It is required that $\mathsf{Dec}_{\mathsf{sk}}\left[\mathsf{Enc}_{\mathsf{pk}}\left(m\right)\right] = m$ except with negligible probability over $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{Gen}\left(\lambda\right)$ and any randomness used by Enc.

In some asymmetric schemes the choice of public key directly influences which messages can be encrypted with the scheme, for example if the public key is a positive integer and the scheme can only encrypt positive integers smaller than the public key.

Hybrid encryption

Public key cryptography did not replace the use of symmetric encryption ciphers. Implementations of public key schemes require the multiplication of very large integers which is much slower than the simple substitutions and exclusive-or operations employed by symmetric ciphers. Asymmetric schemes normally also suffer from ciphertext expansion where the ciphertext is several times larger than the message that was encrypted.

Hybrid encryption is a way of using symmetric and asymmetric schemes together to gain the benefits of both types of schemes. When using hybrid encryption Alice would encrypt a key k using a public key scheme and Bob's public key pk. She would then encrypt her message m using a symmetric cipher with the key k. This is illustrated in Figure 2.1. Now only the symmetric key suffers from ciphertext expansion while the message, that can be much larger, does not and they do not need to transmit over a secure channel to have security.

2.2.2 Perfect Secrecy and Kerckhoff's principle

Perfect secrecy is achievable, as has been proved by Shannon in his seminal paper [35]. The one-time pad, also known as Vernam's Cipher, is such a perfectly secure scheme. Perfect secrecy refers to a cryptographic scheme that is secure against an unbounded attacker while most cryptographic schemes in use today only offer security against polynomially bounded attackers. The one-time pad, however, is of limited practical use. It is a symmetric cipher where the ciphertext is found by calculating the exclusive-or of the key and message. This requires that the size of the key be equal to the size of the message. For the one-time pad to stay perfectly secure, no key may be re-used twice, other than by random chance. This means that the secure channel that has to be established each time to transfer the key could rather have been used to transmit the message itself, making the one-time pad impractical.

Another important idea in modern cryptography is Kerckhoff's principle which states [20]:

> The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.

Another way of saying this is that the only part of the cryptographic scheme that has to be kept secret to ensure that it stays secure, is the key used. One simple reason is that if a key falls into the hands of an enemy it takes a minimal amount of effort to create a different key or even expand the key size, while developing and deploying a whole new cryptographic system is costly. It is also much easier to keep a key secret as it is much smaller than an entire cryptographic system. Releasing the details of a cryptographic scheme also allows a broader audience to inspect the cipher for weaknesses. Additionally, if the encryption method is known, then independent verification of any security claims of the scheme used can be made.

Since perfectly secure cryptographic schemes are not of practical use, what kind of security can a practical scheme offer? In modern cryptography the assumption is made that all adversaries are polynomially bounded. Such an adversary only has storage space and computation time available to it that can be described by some polynomial. Using this assumption the bounds of the security offered by the cryptographic scheme can be explicitly stated.

Security for schemes is proven in one of three ways: explicitly, with mathematical assumptions and by peer review.

2.2.3 Security Proofs

While the one-time pad has an explicit proof of security, this may not be possible for all schemes. Instead, the proof of security is shown to rely on certain assumptions which are widely believed to be true, such as "one-way functions exist" or "no polynomial time

factoring algorithm exists". Basing the security claims on such well-known problems allows us to immediately know that when such a problem is solved, the scheme becomes insecure. Such a proof is constructed by mathematically proving the security to some negligible probability in the security parameter of the scheme.

Other systems rely on extensive peer-review to claim their security. For instance, the Data Encryption Standard (DES) developed in 1977 [37] has no proof of security, but in all the years of scrutiny since its conception, the best known practical attack is a brute force search over the key space. The Advanced Encryption Standard (AES) competition was held to find a successor to DES as the size of its keys have become too small to be considered secure on fast modern hardware. For this competition anyone could submit a candidate cipher which would then be publicly reviewed, including review by other contestants.

In order to understand the security offered by cryptographic schemes it must first be defined what capabilities the attacker has. An adversary's capabilities is composed of both an attack and a security model.

2.2.4 Attack Models

An attack model for an adversary details how an attacker can manipulate and extract information from the cryptographic implementation itself. Attack models for cryptographic implementations can be grouped into three groups [11]:

1. *Black-box:* This is the classical model against which cryptographic implementations are secured. The assumption is this case is that the attacker only has full access to information transmitted over a channel between parties and oracle access to encryption and decryption routines.

2. *Grey-box*: The attacker now has access to data generated by the physical implementation of the cryptographic routine. This includes physical attributes such as power consumption and timing data.

3. *White-box*: In this model it is assumed the attacker has full control over the cryptographic implementation as it executes. Thus the attacker can read the values of intermediate calculations in the implementation as well as modify these values at will.

It should be noted that an attacker operating in the white-box attack model also implicitly operates in the grey and black-box models. Therefore an implementation that is secure in the white-box model is also secure in both the black and grey-box models. Any software implementation of a cryptographic algorithm should be designed within the white-box attack model since they can be executed by an attacker on a host over which they have full control.

Any cryptographic implementation is only secure as long as the key used is not known to the attacker. A cipher secure in the black-box attack model ensures that the attacker cannot learn the key by manipulating the inputs and outputs to the implementation. However, once an attacker gains white-box attack model privileges against a black-box implementation they can simply retrieve the key from the memory of the implementation they are attacking. Thus most implementations secure against the black-box attack model are not at all secure against the white-box attack model.

2.2.5 Security Models

The security model defines the ways in which the attacker can interact on a functional level with the implementation. "On a functional level" refers to which parts of the entire scheme the attacker has access to as it is possible for an attacker to only have access to the encryption routines and not necessarily the decryption ones. If an attacker can receive the decryptions of ciphertexts then we say the attacker has oracle access to the decryption routine. Note that when requesting a decryption and receiving the plaintext from their oracle the attacker does not see the key used, only the result of his inputs.

A useful way of expressing the security model of a scheme is in terms of how indistinguishable the ciphertexts it produces are. The reasoning is that if an attacker cannot even distinguish between the ciphertexts or the ciphertext and a random string then he can learn no information about the plaintext. There are three main security models used for cryptographic schemes [20]:

1. Security in the presence of an eavesdropper,
2. Security against adaptive chosen plaintext attacks (CPA secure), and
3. Security against adaptive chosen ciphertext attacks (CCA secure).

We will now discuss each of these security models in short. The definitions given are only for symmetric encryption schemes as the asymmetric scheme definitions are analogously defined but with the attacker given the public key of the scheme. The security definitions that follows assume that a scheme is broken if an adversary can do significantly better than just trying to guess the answer.

2.2.5.1 Security in the Presence of an Eavesdropper

The first security model has the weakest type of attacker, one who can only see ciphertexts produced by the scheme.

Definition 3. The eavesdropping indistinguishability experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(\lambda)$:

1. A key k is generated by running $\mathsf{Gen}(\lambda)$.

2. The adversary $\mathcal{A}$ is given input λ and outputs a pair of messages m_0, m_1 of the same length.

3. A random bit $b \leftarrow \{0,1\}$ is chosen, and then a ciphertext $\mathsf{c} \leftarrow \mathsf{Enc}_{\mathsf{k}}(m_b)$ is computed and given to $\mathcal{A}$.

4. $\mathcal{A}$ outputs a bit b'.

5. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

This definition can be extended to allow the attacker to query the scheme with a two vectors of messages. One of these vectors is chosen and the messages are encrypted one at a time and returned as another vector. The attacker now has to identify which vector of messages was encrypted by the experiment. Formally security against an eavesdropper can be defined as:

Definition 4. Security in the presence of an eavesdropper

An encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is secure in the presence of an eavesdropper if, for every probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a negligible function negl such that

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda). \tag{2.1}$$

2.2.5.2 Security against Adaptive Chosen Plaintext Attacks

In this security model the attacker gains oracle access to the encryption routine.

Definition 5. The CCA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(\lambda)$:

1. A key k is generated by running $\mathsf{Gen}(\lambda)$.

2. The adversary $\mathcal{A}$ is given input λ and oracle access to $\mathsf{Enc}_{\mathsf{k}}(\cdot)$.

3. $\mathcal{A}$ outputs a pair of messages m_0, m_1 of the same length.

4. A random bit $b \leftarrow \{0,1\}$ is chosen, and then a ciphertext $\mathsf{c} \leftarrow \mathsf{Enc}_{\mathsf{k}}(m_b)$ is computed and given to $\mathcal{A}$.

5. The adversary $\mathcal{A}$ continues to have oracle access to $\mathsf{Enc}_{\mathsf{k}}(\cdot)$.

6. $\mathcal{A}$ outputs a bit b'.

7. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

This definition also has a non-adaptive variant in which step 5 is not present which leads to a less secure scheme. Security against adaptive chosen plaintext attacks is then

defined as [20]:

Definition 6. Security against adaptive chosen plaintext attacks

An encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is secure against adaptive chosen plaintext attacks if, for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function negl such that

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda). \tag{2.2}$$

2.2.5.3 Security against Adaptive Chosen Ciphertext Attacks

In this security model the attacker has the same capabilities as in an adaptive chosen plaintext attack as well as oracle access to the decryption routine of the scheme. We first define the security experiment.

Definition 7. The CCA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(\lambda)$:

1. A key k is generated by running $\mathsf{Gen}(\lambda)$.
2. The adversary $\mathcal{A}$ is given input λ and oracle access to $\mathsf{Enc}_{\mathsf{k}}(\cdot)$ and $\mathsf{Dec}_{\mathsf{k}}(\cdot)$.
3. $\mathcal{A}$ outputs a pair of messages m_0, m_1 of the same length.
4. A random bit $b \leftarrow \{0,1\}$ is chosen, and then a ciphertext $\mathsf{c} \leftarrow \mathsf{Enc}_{\mathsf{k}}(m_b)$ is computed and given to $\mathcal{A}$.
5. The adversary $\mathcal{A}$ continues to have oracle access to $\mathsf{Enc}_{\mathsf{k}}(\cdot)$ and $\mathsf{Dec}_{\mathsf{k}}(\cdot)$, but is not allowed to query the latter on c.
6. $\mathcal{A}$ outputs a bit b'.
7. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

This definition also has a non-adaptive variant in which step 5 is not present which leads to a less secure scheme. Security against adaptive chosen ciphertext attacks is then defined as [20]:

Definition 8. Security against adaptive chosen ciphertext attacks

An encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is secure against adaptive chosen ciphertext attacks if, for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function negl such that

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda). \tag{2.3}$$

2.3 Block and Stream Ciphers

The white-box implementations of two block ciphers is discussed in the next section, but an overview of block and stream ciphers is given here.

Symmetric encryption schemes can be classified as one of two ciphers: stream or block. Stream ciphers make use of the stream of bits generated by a pseudorandom generator and can encrypt messages of length equal to the expansion factor of the generator. Block ciphers are designed with a specific block size and can only encrypt messages of that block size. Block ciphers can be extended to encrypt messages of arbitrary length by running them in different modes of operation, which is discussed later. We first introduce some key concepts in understanding the security behind block and stream ciphers.

2.3.1 One-way Functions and Pseudorandomness

Several cryptographic schemes use one-way functions as a basis of their claims to security. Informally, if a one-way function f and some input x is known, then $f(x)$ is efficiently computable, but if only $f(x)$ and f is known, then x cannot be efficiently calculated. Letting $\{0,1\}^\lambda$ indicate a string of 1's and 0's of length λ, the inverting experiment $\mathsf{Invert}_{\mathcal{A},f}(\lambda)$ with attacker $\mathcal{A}$ against the function f is defined [20]:

Definition 9. The inverting experiment $\mathsf{Invert}_{\mathcal{A},f}(\lambda)$

1. Choose input $x \leftarrow \{0,1\}^\lambda$. Compute $y = f(x)$.
2. $\mathcal{A}$ is given λ and y as input and outputs x'.
3. The output of the experiment is defined to be 1 if $f(x') = y$, and 0 otherwise.

A one-way function is formally defined as [20]:

Definition 10. One-way Function

A function $f : \{0,1\}^* \to \{0,1\}^*$ is one-way if the following two conditions hold:

1. (Easy to compute) There exists a polynomial time algorithm M_f for computing f; that is
$$M_f(x) = f(x); \forall x \in \{0,1\}^*. \tag{2.4}$$
2. (Hard to invert) For every probabilistic polynomial-time algorithm $\mathcal{A}$, there exists some negligible function negl such that the probability
$$\Pr\left[\mathsf{Invert}_{\mathcal{A},f}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda) \tag{2.5}$$
where λ is the security parameter.

A concept similar to one-way functions are trap-door permutations. These permutations are bijective functions that behave as a one-way function so long as a secret trap-door value t stays secret. As soon as t is known, the permutation can be efficiently inverted.

One of the simplest modern cryptographic schemes is a stream cipher that uses a pseudorandom generator for security. Informally a pseudorandom generator outputs a deterministic string such that any adversary who runs in polynomial time is unable to tell the difference between the generated string and a string chosen uniformly at random. Such an adversary is known as a distinguisher, is denoted by the symbol $\mathcal{D}$ and outputs 1 when they think they have been given a truly random string. A pseudorandom generator is thus defined as follows [20]:

Definition 11. Pseudorandom Generator

Let $\ell(\cdot)$ be a polynomial and let G be a deterministic polynomial time algorithm such that for any input $s \in \{0,1\}^{\lambda}$, algorithm G outputs a string of length $\ell(\lambda)$. We say that G is a pseudorandom generator if the following two conditions hold:

1. (Expansion) For every λ it holds that $\ell(\lambda) > \lambda$ and
2. (Pseudorandomness) for all polynomial-time distinguishers $\mathcal{D}$ with $r \leftarrow \{0,1\}^{\ell(\lambda)}$ and $s \leftarrow \{0,1\}^{\lambda}$, there exists a negligible function negl such that

$$\Pr\{[\mathcal{D}(r) = 1] \wedge [\mathcal{D}(G(s)) = 0]\} \leq \mathsf{negl}(\lambda) \tag{2.6}$$

where the s is known as the seed of the pseudorandom generator. The function $\ell(\cdot)$ is called the expansion factor of G.

With some of the basic cryptographic primitives having been presented we can now continue to discuss a few constructions of symmetric cryptographic schemes.

2.3.2 Stream Ciphers

Stream ciphers operate on the same principle as the one-time pad. Take a string of bits and find the exclusive or of it with the message. The one-time pad uses a truly random string whereas stream ciphers use a pseudorandom string. What follows is a formal construction of a stream cipher by using a pseudorandom generator [20].

Definition 12. Stream Cipher

Let G be a pseudorandom generator with expansion factor ℓ. Define a symmetric encryption scheme for messages of length ℓ as follows:

- Gen: On input λ, choose $\mathsf{k} \leftarrow \{0,1\}^{\lambda}$ uniformly at random and output it as the key.
- Enc: on input a key $\mathsf{k} \in \{0,1\}^{\lambda}$ and a message $m \in \{0,1\}^{\ell}$, output the ciphertext $\mathsf{c} = G(\mathsf{k}) \oplus m$.

- Dec: on input a key $\mathsf{k} \in \{0,1\}^\lambda$ and a ciphertext $\mathsf{c} \in \{0,1\}^\ell$, output the ciphertext $m = G(\mathsf{k}) \oplus \mathsf{c}$.

It can be proven that the above construction has indistinguishable encryptions in the presence of an eavesdropper under the assumption that G is a pseudorandom generator. However, this construction does not provide indistinguishable multiple encryptions in the presence of an eavesdropper. This can easily be seen from the fact that the above construction is deterministic. Also, using the construction as described above violates the principle of the one-time pad that no random stream may be reused. A workaround to this is to have the Enc algorithm output a random initializing vector IV together with the ciphertext. The pseudorandom generator then has to be adapted to take both the key and IV as input to produce a pseudorandom stream.

2.3.3 Block Ciphers

A block cipher is an efficient keyed permutation, defined as [20]:

Definition 13. Efficient Keyed Permutation

A function $F : \{0,1\}^\lambda \times \{0,1\}^\ell \to \{0,1\}^\ell$ is an efficient keyed permutation if the following two properties hold

1. The function $F(\mathsf{k}, x)$, referred to as $F_\mathsf{k}(x)$, is a bijection over $\{0,1\}^\ell$ for all $\mathsf{k} \in \{0,1\}^\lambda$.
2. F_k and F_k^{-1} are both efficiently calculable given k.

In the context of a block cipher, we refer to ℓ as the block length and λ as the key length.

The above definition of a block cipher contains no notion of security. The function $F(\mathsf{k}, x) = x; \forall \mathsf{k} \in \{0,1\}^\lambda, x \in \{0,1\}^\ell$ is an efficient keyed permutation, but is not secure in any way. When defining security for pseudorandom permutations the concept of a distinguisher is used as well. When applied to a pseudorandom generator the aim of the distinguisher is to distinguish between a random string and a pseudorandom one. For pseudorandom permutations the distinguisher must distinguish between the pseudorandom permutation and a truly random permutation. It is required that all secure block ciphers be strong pseudorandom permutations, defined as [20]:

Definition 14. Strong Pseudorandom Permutation

Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient, keyed permutation. We say that F is a strong pseudorandom permutation if, for all probabilistic polynomial time distinguishers $\mathcal{D}$, there exists a negligible function negl such that

$$\Pr\left\{\left[\mathcal{D}^{F_\mathsf{k}(\cdot),F_\mathsf{k}^{-1}(\cdot)}(\lambda) = 0\right] \wedge \left[\mathcal{D}^{f(\cdot),f^{-1}(\cdot)}(\lambda) = 1\right]\right\} \leq \mathsf{negl}(\lambda), \tag{2.7}$$

where $\mathsf{k} \leftarrow \{0,1\}^\lambda$ is chosen uniformly at random and f is chosen uniformly at random from the set of all permutations on λ-bit strings.

Thus an attacker must not be able to distinguish between a random permutation and a pseudorandom one, even when given access to both permutations and their inverses.

Modes of Operation for Block Ciphers

Block ciphers are designed to work for a specific block length, but messages will not necessarily fit in the allocated block size. To overcome this limitation there are four commonly used modes of operation in which a block cipher with a block size of ℓ can run to increase the size of the messages it can encrypt. For all modes the message m is split into several blocks $m = m_1 \| m_2 \| \dots \| m_x$, with each m_i of length ℓ. Here $x \| y$ indicates the concatenation of the strings x and y. A padding scheme is required if the message length is not an integer multiple of the block size. A simple padding scheme would be to add zeroes to the left of the message until its length is an integer multiple of the block size. The four modes are [20]

1. Electronic Code Book (ECB) mode (Figure 2.2). Each of the m_i plaintext message blocks are separately passed through the block cipher to produce the ciphertext $\mathsf{c} = F_\mathsf{k}(m_1) \| \dots \| F_\mathsf{k}(m_x)$. This mode of operation is highly insecure as it doesn't even have indistinguishable encryptions in the presence of an eavesdropper, because it is possible that the same block of plaintext can be repeated in the message which would encrypt to the same block of ciphertext which helps an adversary distinguish ciphertexts.

2. Cipher Block Chaining (CBC) mode (Figure 2.3). Here a random initializing vector IV of length ℓ is chosen. This vector is then XORed with the first block of the message before being passed through the block cipher. Each successive block of the plaintext is then first XORed with the ciphertext of the previous block before being passed through the pseudorandom permutation. More concisely, let $\mathsf{c}_0 = \mathsf{IV} \leftarrow \{0,1\}^\ell$ and then for all $i \leq x$, let $\mathsf{c}_i = F_\mathsf{k}(\mathsf{c}_{i-1} \oplus m_i)$. This mode can be proven to be CPA-secure if F is a pseudorandom permutation.

3. Output Feedback (OFB) mode (Figure 2.4). This mode creates a stream cipher by using a block cipher as a pseudorandom generator. This is achieved by taking a random IV of length ℓ and passing it through the block cipher. This is the first block of the random stream. Each block after that is then found by passing the previous block of the random stream through the block cipher. Encryption is achieved by XORing the corresponding message and random blocks. More concisely, let $\mathsf{c}_0 = r_0 = IV \leftarrow \{0,1\}^\ell$ and then for $1 \leq i \leq x$ let $r_i = F_\mathsf{k}(r_{i-1})$ and $\mathsf{c}_i = m_i \oplus r_i$. This mode is also CPA-secure if F is a pseudorandom permutation.

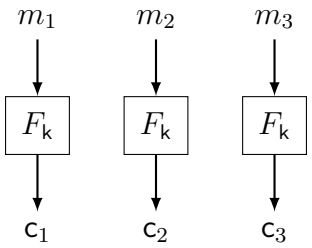


Figure 2.2: Electronic Code Book mode

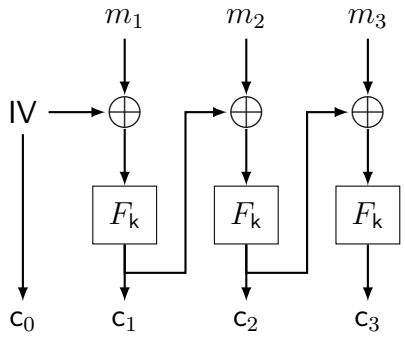


Figure 2.3: Cipher Block Chaining mode

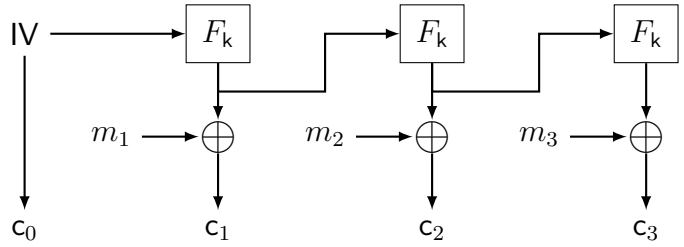


Figure 2.4: Output Feedback mode

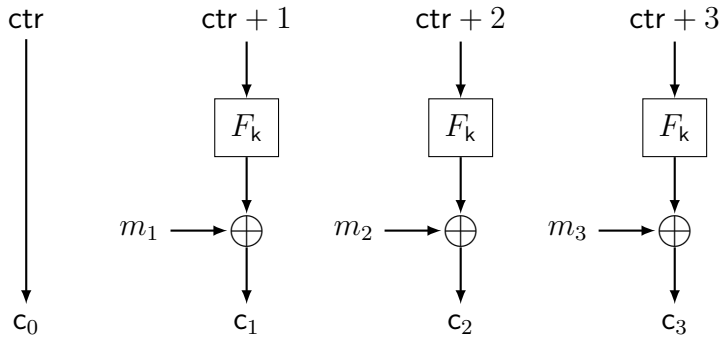


Figure 2.5: Counter mode

4. Counter (CTR) mode (Figure 2.5). This mode also creates a pseudorandom generator by using a block cipher. First a random IV, known as ctr, of length ℓ is chosen. Each block of the random stream is then found by increasing ctr by one and applying the pseudorandom permutation to this new value. The corresponding random blocks and message blocks are then XORed to produce the ciphertext. More concisely, let $\mathsf{c}_0 = \mathsf{ctr} \leftarrow \{0,1\}^\ell$ and then for all $1 \leq i \leq x$ let $\mathsf{c}_i = F_\mathsf{k}(\mathsf{ctr} + i) \oplus m_i$. This mode is CPA-secure but also allows for random access, where the i-th block is decrypted without decrypting any other block of ciphertext.

Constructing Pseudorandom Permutations

It still has not been discussed how to construct a strong pseudorandom permutation. Note that to construct a truly random permutation for strings of length ℓ requires $\ell \cdot 2^\ell$ bits of memory. Such a permutation would be constructed by filling a lookup table with truly random strings of length ℓ. This table would have ℓ entries of size 2^ℓ bits. For $\ell = 32$ this already requires 16 gigabytes of storage. Thus we need to be able to construct a function that is concise and small in its implementation but behaves like a random permutation. There are two popular high-level designs for achieving this: Substitution Permutation Networks and Feistel networks. Both of these are ways of implementing the confusion-diffusion paradigm which was introduced by Shannon [35] to enable the construction of concise, but seemingly random permutations.

The confusion-diffusion paradigm uses small random permutations that can be easily stored to construct larger permutations that seem random. The first step is to introduce confusion into the input. For 128 bits of input this can be achieved by using 16 independent 8-bit permutations. Each of these 16 permutations is applied to a distinct set of 8 bits from the input. The diffusion part is introduced by then re-ordering the bits after it has passed through the permutations. These two steps together are called a round. Rounds are repeated multiple times to introduce more confusion and finally create a pseudorandom permutation.

2.3.3.1 Substitution Permutation Networks (SPN)

Substitution permutation networks are almost a direct implementation of the confusion-diffusion paradigm. The confusion-diffusion paradigm as described above does not introduce any dependency on a key. This dependency is introduced in SPNs by adding an extra step to each round where the intermediate result is XORed with a round key. The round key is derived from the master key, round number and some key schedule. The smaller permutations that are used are known as substitution boxes, or S-boxes for short. According to Kerckhoff's principle the structure of these S-boxes should be publicly known. This does not mean that the S-boxes have to be key-independent. If key-dependent S-boxes

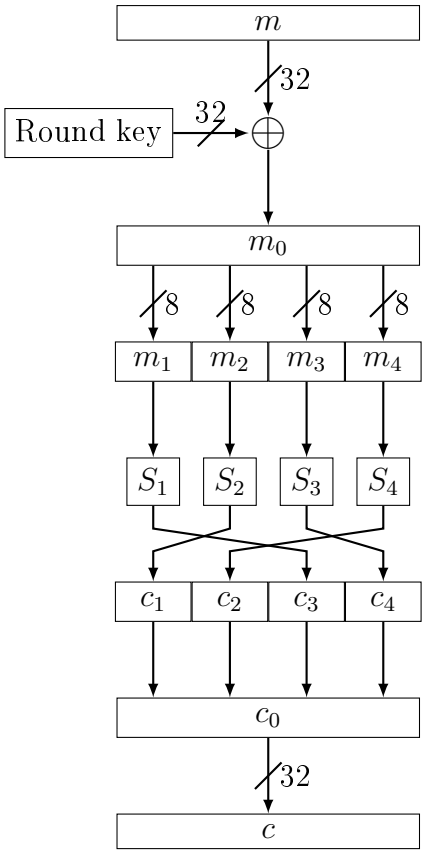


Figure 2.6: Substitution Permutation Network round

are used in a cipher, as in Blowfish [34], the algorithm to construct the S-boxes from the key should still be public. Figure 2.6 shows a round in an SPN with the S-boxes labelled S_1, S_2, S_3, S_4.

SPN designs are based on two important principles:

1. Invertible S-boxes: In order to ensure that the SPN is indeed a permutation it must be invertible. If the S-boxes are invertible it makes each round of the SPN invertible given the key. Since each round can be efficiently inverted this way the SPN is a permutation.

2. Avalanche effect: The avalanche effect aims to introduce large differences in the output of the permutation if small changes are made to the input. If a one bit flip in the input changed only one bit in the output, then the SPN would not look like a random permutation. The avalanche effect can be achieved by making a one bit change in the input to a round result in a two bit change in the output of that round. The mixing permutation at the end of each round then maps the output bits of each S-box to several different S-boxes in the next round. Thus after r rounds, a one bit change in the input results in an approximately 2^r bit change of the output.

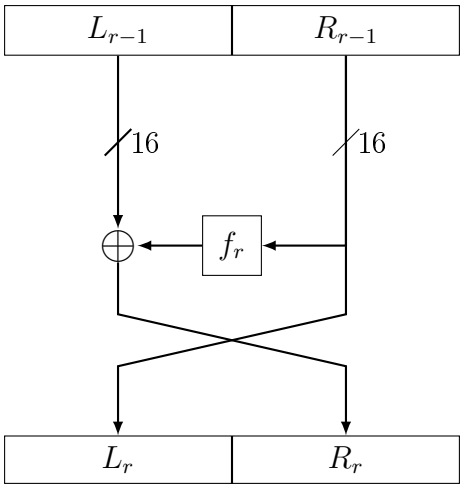


Figure 2.7: Feistel Round

2.3.3.2 Feistel Networks

A Feistel network does have not the design requirement of an SPN that the S-boxes used must be invertible. A Feistel network's S-boxes are incorporated a round mangler function, denoted f_r for round r. The round mangler function also contains some operation that is dependent on the round key which is derived from the encryption key, round number and some key schedule. In its basic high-level design a Feistel network round, depicted in Figure 2.7, works as follows [20]

1. The input to a round r is split into two halves, L_{r-1} and R_{r-1}.
2. If the round mangler function is $f_r(x)$ then define $R_r = L_{r-1} \oplus f_r(R_{r-1})$ and $L_r = R_{r-1}$.
3. The output of a round is defined as the concatenation of L_r and R_r.

Although the S-boxes used in a Feistel network are not invertible, a Feistel network is still a permutation since given the output $L_r \| R_r$ to some round r, it can be shown that each round is invertible as follows:

1. Set $R_{r-1} = L_r$ and
2. calculate $R_r \oplus f_r(L_r) = L_{r-1} \oplus f_r(R_{r-1}) \oplus f_r(L_r) = L_{r-1} \oplus f_r(R_{r-1}) \oplus f_r(R_{r-1}) = L_{r-1}$.

Thus each round of a Feistel network can be inverted given the round mangler function and the output of that round.

We discussed the basic design principles and high-level designs of two categories of symmetric encryption schemes. Both Feistel and Substitution Permutation Networks were discussed as well as the modes of operation for a block cipher. This background will be useful in understanding the techniques described in the next section regarding white-box cryptography.

2.4 White-box Cryptography

Introduction

In the seminal papers of Chow et al. [10, 11] the white-box attack context was introduced. Their definition of this attack model is:

Definition 15. The White-box Attack Context

The attacker is assumed to have all the advantages of an adaptive chosen ciphertext attack, plus full access to the encrypting software and control of the execution environment. This includes arbitrary trace execution, examining sub-results and keys in memory, performing arbitrary static analyses on the software, and altering results of sub-computation (e.g. via breakpoints) for perturbation analysis.

An implementation of a cryptographic cipher is said to be white-box if it is secure in the white-box attack context. Chow et al. motivate their consideration of this attack model by the spread of commercial applications that employ cryptography to protect content onto commodity hardware that is untrusted. Their work can also be seen as an extension of previous work such as server-aided RSA [24, 8, 36]. The goal of server-aided RSA is to enable the use of computationally heavy cryptographic schemes such as RSA on secure hardware that is computationally weak. In this model many computations are done by a second processor that is untrusted but computationally powerful. White-box cryptography aims to do the entire cryptographic calculation on the more powerful but untrusted processor without compromising the security of the cryptographic implementation.

Chow et al. present practical white-box implementations of both DES [11] and AES [10]. The implementations proposed rely on transforming the underlying functions of the block cipher into a network of lookup tables and hiding the key inside these tables in such a way that an attacker cannot extract the key.

The Implementation of Chow et al.

The primary aim of the white-box implementations described by Chow et al. is to make key recovery impossible. The key they refer to here is the original AES or DES key that was used to create the implementation. Note that once the key has been chosen for either of these schemes, the S-boxes and key mixing operations for each round is fixed. This fact is exploited in their implementation to change all the S-boxes and key mixing operations into a set of key-dependent S-boxes. This unfortunately means that the same S-boxes cannot be reused for each round and the implementation has a significantly larger size. These new key-dependent S-boxes are then broken up onto a network of lookup tables.

The construction of the original S-boxes are publicly known which makes key recovery from the corresponding key-dependent implementations trivial. To counter this, they introduce the idea of mixing bijections. The idea is that if $S(\cdot)$ is some key-dependent

S-box then two random bijections F and G are defined with the appropriate input size and attached to $S(\cdot)$ in the following way: $F^{-1} \circ S(\cdot) \circ G$. This ensures that all the key-dependent S-boxes have local security, meaning that these encoded S-boxes themselves do not leak any information about the key they contain. This is because the input and output of the entire lookup network for an encoded S-box can stay the same even if the S-box in the middle is changed by adapting F and G. If in the original implementation two S-boxes had to be evaluated after each other such as $S_1(\cdot) \circ S_2(\cdot)$ then with the mixing bijections it becomes $(F^{-1} \circ S_1(\cdot) \circ G) \circ (G^{-1} \circ S_2(\cdot) \circ H)$. Thus the input mixing bijection to $S_2(\cdot)$ is cancelled out by the output mixing bijection of $S_1(\cdot)$. While these mixing bijections provide local security for each S-box, they only force the attacker to inspect a greater amount of the implementation to recover the key. The S-boxes are also shuffled so that the attacker cannot be sure which S-box is implemented in a certain set of lookup tables.

Cryptanalysis

Both of the white-box implementations presented by Chow et al. have been successfully broken in [3, 17, 40]. Of note is the claim by Wyseur in [39] that block ciphers using Feistel rounds are susceptible to the cryptanalysis presented due to their inherent structure design which only modifies half of the input to each round. It is unclear whether this attack is aided by the fixed S-boxes of DES and if other Feistel round ciphers that use key-dependent S-boxes, such as Blowfish [34], will be as susceptible to cryptanalysis.

An attack has been demonstrated in [40] in which they exploit non-random propagation of plaintext changes in the structure of the the DES implementation. These propagations are used to identify S-boxes and the inputs to these S-boxes. The key is then recovered by guessing a bit and then checking the effect of this guess to determine one other bit of the key. This attack finally gives two complementary keys that are both valid due to the complementation property of DES.

The methods discussed so far have all required that both parties attempting to communicate have access to some shared key. In the following sections we will discuss methods in which two parties can agree on a key to use without having access to a secure channel for transferring the key.

2.5 Public Key Cryptography

Even though public key cryptography offers an attractive advantage over symmetric schemes, it has not replaced the use of symmetric schemes. Most asymmetric schemes suffer from ciphertext expansion, where the ciphertext is larger by a factor of 2 or more than the message that is being encrypted. The exponentiation of large integers used by

asymmetric schemes also takes longer than the substitution rounds used by symmetric schemes. Public key schemes allow for the creation of a secure channel between two parties through which a random symmetric key can be transmitted when only an unsecured channel is available to them. The rest of the message is then encrypted using this symmetric key together with a symmetric scheme. We present two popular and standardized public key schemes here.

Before we introduce some practical asymmetric schemes a short discussion on groups and number theory is given which is required in order to understand how the presented schemes work.

2.5.1 Number Theory and Groups

To understand the way that security is proven for asymmetric encryption schemes the concept of a abelian group must first be introduced. Asymmetric schemes can be implemented using any underlying abelian group but they can only be proven secure when using certain groups. The groups that give security to these schemes depend on which problems the scheme assumes are unsolvable in polynomial time. For an abelian group a binary operation on a set G is a function that takes as input two elements from the set G. Formally an abelian group is defined as [20]:

Definition 16. Abelian Group

A set G together with a binary operation $\circ$ is an abelian group if the following properties all hold

1. (Closure) For all $g, h \in G$ it holds that $g \circ h \in G$.
2. (Identity element) There exists an identity element $e \in G$ such that for all $g \in G$, $e \circ g = g = g \circ e$.
3. (Inverse) For any $g \in G$ there exists an element $h \in G$, known as the inverse of g, such that $g \circ h = e = h \circ g$.
4. (Associativity) For all $g_1, g_2, g_3 \in G$ it holds that $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.
5. (Commutativity) For all $g, h \in G$, $g \circ h = h \circ g$.

For a set and a binary operation to be classified as a group all these properties except commutativity have to hold. If a abelian group has a finite number of elements then it is known as a finite abelian group. We let $|G|$ denote the number of elements in a finite group, also known as its order.

It can be shown that a group has a unique identity element and each element in the group has a unique inverse. A group can be described using either additive or multiplicative notation. For this thesis we will use the latter. With multiplicative notation the

group operation applied to two elements g, h is denoted $g \cdot h$ or gh, the inverse of g is denoted by g^{-1} and the identity element is denoted by 1. Repeated application of the group operation will be denoted by exponentiation: $g \cdot g \cdot g = g^3$. From these definitions the following can be proven [20]

Lemma 17. *Let G be a group and $a, b, c \in G$. If $ac = bc$ then $a = b$. In particular, if $ac = c$ then a is the identity element of G.*

Proof. Suppose it is known that $ac = bc$. Multiplying both sides with the unique inverse c^{-1} of c, we obtain $a = b$:

$$ac = bc \Rightarrow (ac) \cdot c^{-1} = (bc) \cdot c^{-1} \Rightarrow a(cc^{-1}) = b(cc^{-1}) \Rightarrow a = b. \tag{2.8}$$

□

Theorem 18. *Let G be a finite abelian group with $\ell = |G|$, the order of the group. Then for any element $g \in G$, $g^\ell = 1$.*

Proof. Choose an arbitrary $g \in G$ and let $g_1, \ldots, g_\ell$ be the elements of G. We claim that

$$g_1 \cdot g_2 \cdots g_\ell = (gg_1) \cdot (gg_2) \cdots (gg_\ell). \tag{2.9}$$

To see this, note that $gg_i = gg_j$ implies $g_i = g_j$ by Lemma 17 therefore each of the ℓ elements in parentheses on the right-hand side of (2.9) is unique. Because there are exactly ℓ elements in G the product on the right-hand side is simply the elements of G being multiplied together in some permuted order. Since G is associative the order of multiplying the elements together does not matter and thus the left-hand side is equal to the right-hand side.

Using the fact that G is abelian we can write

$$g_1 \cdot g_2 \cdots g_\ell = (gg_1) \cdot (gg_2) \cdots (gg_\ell) = g^\ell \cdot (g_1 \cdot g_2 \cdots g_\ell) \tag{2.10}$$

which by Lemma 17 implies that $1 = g^\ell$. □

Using this theorem the following corollary can be proved which allows for the exponent to be reduced modulo the group order. This corollary is later used to prove the correctness of the RSA encryption scheme.

Corollary 19. *Let G be a finite group with $\ell = |G| > 1$. Then for any $g \in G$ and any integer i it holds that $g^i = g^{i \pmod \ell}$.*

Proof. Let $i = q\ell + r$ with $q, r \in \mathbb{Z}$ and $r = i \pmod \ell$. Using Theorem 18,

$$g^i = g^{q\ell + r} = g^{q\ell} \cdot g^r = \left(g^\ell\right)^q \cdot g^r = 1^q \cdot g^r = g^r, \tag{2.11}$$

as claimed. □

Some asymmetric schemes use the concept of a cyclic group [20]:

Definition 20. Cyclic Group and Generators

A group G with $|G| = \ell$ is said to be cyclic if there exists a $g \in G$ for which the sequence $g, g^2, g^3, \ldots, g^\ell$ contains every element in G. Such a g is called a generator of G.

Proposition 21. *Let G be a cyclic group with $|G| = \ell$ and g a generator of G. Then for any $x, y \in \mathbb{Z}$ we have that $g^x = g^y$ if and only if $x = y \pmod{\ell}$.*

Proof. If $x = y \pmod{\ell}$ then $x \pmod{\ell} = y \pmod{\ell}$ and from Corollary 19 it holds that

$$g^x = g^{x \pmod{\ell}} = g^{y \pmod{\ell}} = g^y. \tag{2.12}$$

For the other direction we have $g^x = g^y$ and let $x' = x \pmod{\ell}, y' = y \pmod{\ell}$. Again from Corollary 19 it holds that $g^{x'} = g^{y'}$ or equivalently that $g^{x'} \left(g^{y'}\right)^{-1} = 1$. If $x' \neq y'$ it can be assumed without loss of generality that $x' > y'$. Since both x' and y' are smaller than ℓ, the difference $x' - y'$ is then a non-zero integer smaller than ℓ. But then

$$1 = g^{x'} \left(g^{y'}\right)^{-1} = g^{x'-y'}, \tag{2.13}$$

which means the sequence $g, g^2, g^3, \ldots, g^\ell$ will be equivalent to

$$g, g^2, g^3, \ldots, g^{x'-y'-1}, 1, g, \ldots, g^\ell \tag{2.14}$$

and contain at most $x' - y' < \ell$ elements of G contradicting the claim that g is a generator of G. □

Having discussed the required concepts two practical public key encryption schemes will now be introduced.

2.5.2 RSA

Rivest, Shamir and Adleman published one of the first public key schemes. They developed the RSA algorithm in 1978 [20]. If N is the product of two distinct primes, also known as a semiprime, then the RSA encryption scheme encrypts integers chosen from the set $\mathbb{Z}_N^*$. $\mathbb{Z}_N^*$ is the group of all integers smaller than N which is also relatively prime to it and is an abelian group under multiplication modulo N. The order of this group is given by the Euler totient function, $\phi(N)$. Because N is the product of two primes p and q, it is known that $\phi(N) = (p-1)(q-1)$. The algorithm is still practical today if large enough primes are used. The security of this scheme relies on the RSA assumption. First

define $\mathsf{GenRSA}(\lambda)$ as follows:

Definition 22. $\mathsf{GenRSA}(\lambda)$

1. Choose primes p and q, with $\|p\| = \|q\| = \lambda$, and calculate $N = pq$.
2. Find e such that $\gcd(e, \phi(N)) = 1$.
3. Determine $d = e^{-1} \pmod{\phi(N)}$.

Then we define the security experiment $\mathsf{RSA} - \mathsf{inv}_{\mathcal{A},\mathsf{GenRSA}}(\lambda)$:

Definition 23. The RSA experiment $\mathsf{RSA} - \mathsf{inv}_{\mathcal{A},\mathsf{GenRSA}}(\lambda)$

1. Run $\mathsf{GenRSA}(\lambda)$ to obtain (N, e, d).
2. Choose a random $y \leftarrow \mathbb{Z}_N^*$.
3. The attacker $\mathcal{A}$ is given (N, e, y) and outputs $x \in \mathbb{Z}_N^*$.
4. The output of the experiment is 1 if $x^e = y \pmod{N}$.

And finally we can define the RSA hardness assumption:

Definition 24. RSA Hardness Assumption

The RSA problem is hard relative to GenRSA if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, there exists a negligible function negl such that

$$\Pr\left[\mathsf{RSA} - \mathsf{inv}_{\mathcal{A},\mathsf{GenRSA}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda). \tag{2.15}$$

The RSA assumption is that there exists a GenRSA for which it is computationally infeasible to find $y^{e^{-1}} \pmod{N}$ without knowing the value of $\phi(N)$.

To see that d is the solution to the problem we apply Corollary 19. Thus if $x = y^d$, then

$$x^e = \left(y^{d \pmod{\phi(n)}}\right)^e = y^{e^{-1}e} = y \pmod{N}. \tag{2.16}$$

In order to find $e^{-1} \pmod{\phi(N)}$ an attacker must determine $\phi(N)$. It is believed to be computationally infeasible to find $\phi(N)$ from N since no polynomial time algorithm has yet been found for factoring an integer and these problems can be shown to be equivalent. This does not mean that there are no other attack vectors to the RSA assumption, only that the RSA problem is at most as hard as factoring.

There are several known attacks on the direct implementation of the RSA problem as a cryptographic scheme when choosing y as the message to be encrypted. A direct implementation is deterministic and thus vulnerable to chosen plaintext attacks. This

leads to the development of padded RSA which is the implementation that should be used [20]:

Definition 25. Padded RSA

Let GenRSA be as before and let ℓ be a function with $\ell(\lambda) < 2\lambda - 2$ for all λ. Define a public-key encryption scheme as

- Gen: on input λ, run $\mathsf{GenRSA}(\lambda)$ to obtain (N, e, d). Output the public key $\mathsf{pk} = (N, e)$ and the private key $\mathsf{sk} = (N, d)$.
- Enc: on input a public key $\mathsf{pk} = (N, e)$ and a message $m \in \{0,1\}^{\ell(\lambda)}$, choose a random string $r \leftarrow \{0,1\}^{\|N\|-\ell(\lambda)-1}$. Because all positive integers have a binary representation the concatenated string $r\|m$ can be interpreted as an element of $\mathbb{Z}_N$. Calculate and output the ciphertext

$$\mathsf{c} = (r\|m)^e \pmod{N}. \tag{2.17}$$

- Dec: on input a private key $\mathsf{sk} = (N, d)$ and a ciphertext $\mathsf{c} \in \mathbb{Z}_N^*$, compute

$$\hat{m} = \mathsf{c}^d \pmod{N}, \tag{2.18}$$

and the binary representation of $\hat{m}$ will be equal to the string $r\|m$. Outputting the $\ell(\lambda)$ low-order bits of $\hat{m}$ gives the original message m.

The added random bits prohibits the attacker from executing a chosen plaintext attack. Also, half of the ciphertext is used to encrypt the random data, leading to the ciphertext expansion.

While the RSA encryption scheme relies solely on the RSA assumption and must be implemented over the integers other public key schemes exists such that it works for several different underlying groups. El Gamal is one such scheme.

2.5.3 El Gamal

The El Gamal encryption scheme is a public key encryption scheme that allows for some flexibility toward which underlying group is used in its implementation. This is different from RSA where only the group $\mathbb{Z}_N^*$ works.

If Gen is a polynomial time algorithm that on input λ generates a group and returns its description G, the order of the group q, with $\|q\| = \lambda$, and g a generator of the group, the El Gamal encryption scheme is defined as follows:

Definition 26. The El Gamal encryption scheme

- Gen: on input λ run $\mathsf{Gen}(\lambda)$ to obtain (G, q, g). Then choose a random $x \leftarrow \mathbb{Z}_q$ and compute $h = g^x$. The public key is (G, q, g, h) and the private key is (G, q, g, x).

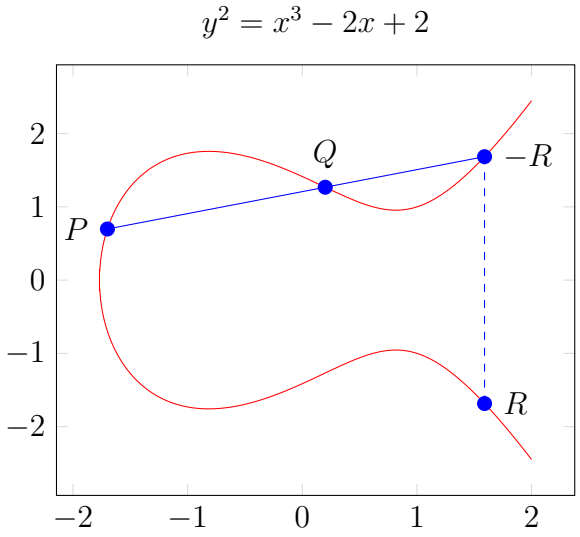


Figure 2.8: An example Weierstraß curve over $\mathbb{R}$

- Enc: on input a public key $\mathsf{pk} = (G, q, g, h)$ and a message $m \in G$, choose a random $y \leftarrow \mathbb{Z}_q$ and output the ciphertext

$$\mathsf{c} = (g^y, h^y \cdot m)\,. \tag{2.19}$$

- Dec: on input a private key $\mathsf{sk} = (G, q, g, x)$ and a ciphertext $(\mathsf{c}_1, \mathsf{c}_2)$, output

$$m = \frac{\mathsf{c}_2}{\mathsf{c}_1^x}. \tag{2.20}$$

Decryption works correctly because $\frac{\mathsf{c}_2}{\mathsf{c}_1^x} = \frac{h^y \cdot m}{(g^y)^x} = \frac{(g^x)^y \cdot m}{g^{xy}} = m$. This encryption scheme has indistinguishable encryptions under a chosen plaintext attack given that the Decisional Diffie Hellman (DDH) problem is hard relative to G [20]. The DDH problem can be defined as [20]:

Definition 27. The Decisional Diffie Hellman problem

Given a cyclic group G and a generator $g \in G$, define $DH_g(h_1, h_2) = g^{log_g h_1 \cdot log_g h_2}$. The DDH problem is to distinguish $DH_g(h_1, h_2)$ from a random group element with non-negligible probability for a randomly chosen h_1 and h_2.

One group for which it is believed that the DDH problem is hard is the group of points on an elliptic curve. The group operation for points on an elliptic curve is known as addition but is very different from addition of integers or vectors. Adding two points on an elliptic curve is described in more detail in the next section.

2.5.4 Elliptic Curves and Bilinear Maps

An elliptic curve is defined over the finite field $\mathbb{F}_p$ and can be represented by the Weierstraß equation $y^2 = x^3 + ax + b$ with $a, b \in \mathbb{F}_p$ and p a prime [22]. There is an additional requirement that $4a^3 + 27b^2 \neq 0$. This requirement ensures that there exists a tangent at every point on the curve. The elliptic curve, known as $E(\mathbb{F}_p)$, is then the set of points $(x, y) \in \mathbb{F}_p^2$ that satisfy the Weierstraß equation $y^2 = x^3 + ax + b$ together with a special point known as $\mathcal{O}$. This point is defined to be at infinity.

$E(\mathbb{F}_p)$ has a group structure with $\mathcal{O}$ being the identity element of the group. The binary operation of the group is commonly described as addition and has a simple geometric interpretation. To sum two points in $E(\mathbb{F}_p)$ a line is drawn through them and the third intersection the line makes with the curve is taken as the negative of the answer. Since the curve is symmetrical around the x-axis there exists a point on the curve that has the same x coordinate but the negative of the y coordinate. This point is taken as the sum of the previous two points. This is shown in Figure 2.8.

For the three points $P(x_P, y_P), Q(x_Q, y_Q), R(x_R, y_R)$ with $P + Q = R$ this group operation can be defined as

$$x_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q, \; y_R = \frac{y_Q - y_P}{x_Q - x_P}(x_P - x_R) - y_P. \tag{2.21}$$

This definition does not work if $P = Q$ since then the denominator $(x_Q - x_P)$ would be zero. If $P = Q$ the following definition for the group operation is used instead

$$x_R = \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P, \; y_R = \frac{3x_P^2 + a}{2y_P}(x_P - x_R) - y_P. \tag{2.22}$$

When using a elliptic curve a base point $g \in \mathbb{F}_p$ is chosen and all other points are simply multiples of this point. Note that g is not necessarily a generator for $E(\mathbb{F}_p)$ but is the generator of a subgroup. The order of this subgroup is usually chosen to be prime when using elliptic curves for cryptographic purposes.

Elliptic curves also allow for a construction known as a bilinear map, which we shall discuss next.

Bilinear Maps

Definition 28. Bilinear Map

Let G_1, G_2 and G_T be cyclic groups that have the same order. A bilinear map $e : G_1 \times G_2 \to G_T$ is a mapping such that for all $u \in G_1$, $v \in G_2$ and $a, b \in \mathbb{Z}$ it holds that

$$e(u^a, v^b) = e(u, v)^{ab}. \tag{2.23}$$

From this definition the following lemma can be proven:

Proposition 29. *If g is a generator of G_1, $G_1 = G_2$ and $u = g^a, v = g^b, w = g^c$ for any $a, b, c \in \mathbb{Z}$ it holds that $e(uv, w) = e(u, w) \cdot e(v, w)$.*

Proof.

$$\begin{aligned} e(uv, w) &= e(g^a g^b, g_1^c) \\ &= e(g, g)^{ac+bc} \\ &= e(g, g)^{ac} \cdot e(g, g)^{bc} \\ &= e(g^a, g^c) \cdot e(g^b, g^c) \\ &= e(u, w) \cdot e(v, w). \end{aligned} \tag{2.24}$$

□

Definition 28 allows for degenerate maps. A degenerate bilinear map takes any pair of inputs and outputs the identity element of the group G_T. To address this issue an admissible bilinear map is defined.

Definition 30. Admissible Bilinear Map

Let $e : G_1 \times G_2 \to G_T$ be a bilinear map and g_1 a generator of G_1 and g_2 a generator of G_2. e is said to be an admissible bilinear map if $e(g_1, g_2)$ outputs a generator of G_T and e is efficiently computable.

Bilinear maps on elliptic curves were originally introduced by Weil [27] in an attempt to break elliptic curve cryptographic systems. Recall the Decisional Diffie-Hellman problem stated in Definition 27. To solve the DDH an attacker must determine if $g^c = g^{ab}$ for random $a, b \in \mathbb{Z}$ when given g, g^a, g^b, g^c. With a bilinear map e this is easy, as an attacker can find $e(g^a, g^b) = e(g, g)^{ab}$ and check for equality against $e(g, g^c) = e(g, g)^c$. If $e(g, g)^{ab} = e(g, g)^c$ then Proposition 21 implies that $c = ab$ and thus $g^c = g^{ab}$.

Bilinear maps have since been used as a primitive for the construction of cryptographic schemes and has even found their way into the field of broadcast cryptography where they have been used to construct bandwidth-efficient schemes.

2.6 Broadcast Encryption

2.6.1 Overview

The term broadcast encryption was first used by Fiat and Naor in [16]. In their paper they define that the goal of a broadcast encryption scheme is to allow an arbitrary set of receivers chosen by a broadcasting centre to decrypt a secured transmission. Some of the

viewers in the total viewer population might be dishonest and attempt to gain access to broadcasts for which they are not part of the entitled set. These dishonest viewers can pool their knowledge and resources to form a collusion with the intent of breaking the security of the scheme employed by the broadcast centre. Fiat an Naor define a broadcast encryption scheme to be k-resilient if it secure against any collusion of k dishonest viewers out of a total of n viewers.

The physical environment in which a broadcast encryption scheme is implemented places constraints on the amount of resources available for the scheme to use. Fiat and Naor identify three categories of computational resources that must be considered and which they attempt to optimize for the design and implementation of a broadcast encryption scheme:

- The number of key management message headers,
- the number of keys associated with each viewer and
- the computational effort required to derive a shared session key.

The importance of each category depends on the environment in which the scheme will be implemented.

Subsequent research into broadcast encryption presents the idea of permanent key revocation. A temporary revocation is when a viewer is not in the entitled group for a specific broadcast but may be included in the entitled set for some future broadcasts. Permanent revocation happens when a viewer is identified as being dishonest and denied access to all future broadcasts. The keys assigned to this dishonest viewer cannot be reassigned to a new viewer and the broadcast centre has the total viewer population it can service reduced by one. Newer schemes address this problem by updating the keys of all honest viewers and creating a new key in place of the dishonest viewer's key.

The message header of any broadcast encryption scheme includes a description of the entitled set. The increase of size in the message header that can be attributed to this set description is ignored when analysing the bandwidth requirements of a particular scheme as it is the same for all schemes, namely a simple bit vector with the indexes of the set bits correlating with the entitled viewers.

2.6.2 Prior Work

Fiat and Naor [16] first present a k-resilient scheme without any key management transmissions. This scheme uses no cryptographic assumptions to prove its security. Unfortunately this scheme requires that each receiver must store $\binom{n}{k}$ keys, which increases exponentially in k. In the case where $k = 1$ this scheme requires each receiver to store n keys. They improve the efficiency for this case by using the assumption that one-way

functions exist and hence pseudorandom number generators can be constructed. These pseudorandom number generators are then used to create a binary tree with the leaves of the tree representing the keys in the system. This approach allows for a 1-resilient scheme while storing only $O(log(n))$ keys. This scheme is broken when two receivers collude because between them they have access to all the keys in the system.

The next scheme they present requires key management transmission overhead. They use 1-resiliant schemes to construct k-resilient schemes. The best efficiency they claim is that the key management transmission overhead will be $O(k^2log^2(k)log(n))$ and the number of keys that have to be stored at each receiver is $O\left(k \cdot log(k)log(n)\right)$. It is interesting to note that the transmission overhead, as well as the number of keys that have to be stored, is independent of the number of privileged users.

The publication of the paper by Fiat and Naor generated much interest in broadcast encryption and many schemes have been proposed to solve the problem. These newer schemes are unfortunately designed for one of two specific cases: either there are many privileged users or there are very few privileged users. This means that for a very dynamic set of privileged users two broadcast encryption systems have to be present to minimize transmission costs. While these schemes are not useful in a very dynamic privileged set environment, they do present some interesting ideas. The first approach of these schemes is to encrypt a symmetric session key several times using different keys. These encrypted session keys constitute the key management transmission overhead. If a receiver is in a privileged set, it can derive a key that will be able to decrypt one of the transmitted encrypted session keys which enables them to decrypt the message.

Naor and Lotspeich published two schemes in 2001 [30] that they call the complete subtree and subtree difference methods. These schemes are very similar to the 1-resiliant scheme proposed by Fiat and Naor [16] that uses binary trees. The schemes assign each receiver to a leaf in a binary tree. The first scheme then finds the minimum number of subtrees needed to exclude all the non-privileged receivers and uses the keys assigned to the roots of these subtrees to encrypt the session key. The second scheme finds subtrees where each subtree contains some revoked receivers, but these revoked receivers make up all the leaves of another subtree. Their schemes require a transmission overhead near linear in the number of revoked receivers and the number of keys stored per receiver is logarithmic.

Jho creates a scheme based on p-punctured c-intervals in [18]. A p-punctured c-interval is an interval of at most c receivers with p or fewer revoked receivers in the interval. The idea is that each receiver can build key chains using one-way permutations. All receivers know all the one-way permutations assigned to their interval. Receivers cannot generate keys for an interval which they are not part of. Their scheme makes the transmission overhead grow linearly in the number of revoked users.

Boneh, Gentry and Waters [4] propose a scheme based on asymmetric encryption that

does not encrypt the same key multiple times under different keys. The values sent as the transmission overhead is used in conjunction with a receiver's private key and the system's public key to calculate the session key. To achieve this they use a bilinear map to pair elements of multiplicative cyclic groups and derive the session key. For each receiver in the system two group elements have to be included in the public key of the system, which has to be stored on all receivers. To achieve efficiency in their scheme they propose to run A systems in parallel, each servicing B receivers for a total of $AB = n$ receivers. An extra group element is then included in the transmission overhead so that receivers of different instances of schemes can still derive a common session key. The reason for creating multiple instances in parallel is that it allows for flexibility in the size of the public key, which is linear in the total number of users for an instance. Thus the transmission overhead will include $A + 1$ group elements and the public key will contain $2B + A$ elements. If $A = B$ then both the public key and transmission overhead will be $O(\sqrt{n})$.

Junod proposes an attribute-based encryption scheme in [19] that can evaluate AND, OR and NOT clauses. By setting the attributes for each transmission correctly it can be used as a product entitlement system. Their system is an extension of the Boneh, Gentry and Waters system described above. They achieve transmission overhead of $O(\alpha\psi)$ where α is the number of clauses, when writing a policy in disjunctive normal form or conjunctive normal form, and ψ is the maximum number of attributes per clause. These are attributes that the receiver has, such as the firmware version number or geographical location. This gives similar transmission overhead to the scheme it is based on, but with more flexible access policies.

Delerabelée, Paillier and Pointcheval propose another scheme based on bilinear maps in [12]. Their scheme achieves better efficiency than the Boneh, Gentry, Waters scheme [4] by having constant size public keys and private keys and a transmission overhead linear in the number of revoked users. It differs from the Gentry scheme by requiring more divisions in the cyclic group. They also give a method to revoke keys in the long term by broadcasting two group elements and a hash. This has the benefit that if a compromised key is found in a pirate network then it can be removed from the system entirely without burdening the transmission overhead for each broadcast. Their scheme can be run in different modes, one of which offers $O(1)$ transmission overhead but requires more storage.

We looked at the origin of broadcast encryption and several schemes to facilitate the efficient entitlement of a privileged set of users. These broadcast encryption schemes studied here are all resilient to a collusion of users outside the privileged set, but do not offer protection against extracting a key from the receiver itself. To protect against this other methods of protecting the implementations of the algorithms have to be looked at. This is where the field of white-box cryptography can be useful.

2.7 The workings of a Set Top Box

When designing a product entitlement system for practical use the environment in which the system will be deployed must be taken into account. Since satellite broadcasting networks are an example of a bandwidth constrained environment, a overview of how a typical product entitlement system works for satellite TV networks will be given here.

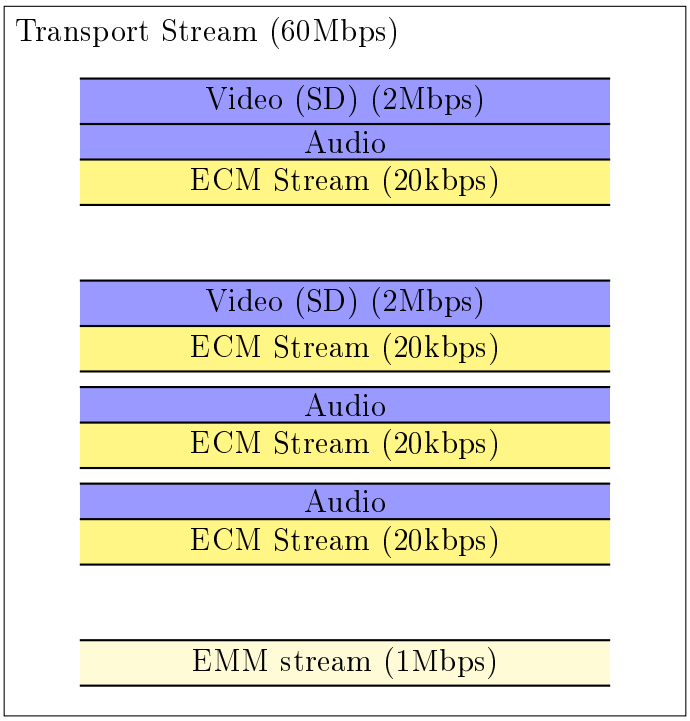


Figure 2.9: A typical Transport Stream

For satellite TV networks the television set itself cannot decrypt the broadcasted content. To facilitate this decryption step each viewer is issued a set top box (STB). Satellite TV networks operate in a pure broadcasting environment which means the STB has no mechanism to send any information back to the broadcaster. The STB receives all its information through a transport stream (TS). The breakdown of a typical TS is shown in Figure 2.9 [13]. In this figure blue boxes indicate content that is directly consumed by the end viewer and the yellow boxes represent the information needed to enable the product entitlement system in use to work correctly. Since the TS is limited to a certain amount of bandwidth it is in the best interest of the broadcaster to have as little as possible of this bandwidth taken up by product entitlement information since the viewer will only care about the data they can consume.

As can be clearly seen from Figure 2.9, there are two different types of entitlement information streams inside an TS: the entitlement control message (ECM) stream and the entitlement management message (EMM) stream. An ECM stream is bound to a specific set of audio and video streams while an EMM stream is not associated with any

specific data stream. The ECM stream contains information necessary to decrypt its associated audio or video stream. An ECM stream is normally around 20kbps which is quite small when compared to the 2Mbps that the video stream requires. Because of this each video and audio stream can have their own ECM stream associated with them without influencing the overall bandwidth requirements by much.

The EMM stream contains the information required to access the contents of the ECM stream. A satellite broadcaster typically offers a large variety of channels, too many to fit into a single TS and thus broadcast their content across several different TSs at the same time. Unfortunately an STB can only tune into one TS at a time and can therefore only access the EMM stream in the TS it is tuned to. This presents a problem when a viewer changes to a channel in a different TS since now the STB has to wait for the necessary information to be sent across the EMM stream so that it can access the associated ECM stream. To address this problem the EMM stream is replicated across all TSs that a broadcaster is using. The EMM stream then contains the information needed by an STB to access any ECM stream if that receiver is in the privileged set that is allowed to do so. The EMM stream is given more bandwidth than each of the ECM streams because the EMM stream is associated with all subscribers of the network. Due to the broadcast nature of satellite networks the broadcaster has no feedback whether or not an STB has received a specific EMM. The STB might have been switched off or be experiencing bad reception. This requires the broadcaster to repeatedly send EMMs to ensure that all STBs receive them. Because an EMM stream is chosen to be around 1Mbps, in a network of 3.5 million subscribers (like DStv has in South Africa [29]), it would take 15 minutes of faultless transmission to broadcast a single EMM for each STB.

We have described the TS that is received by the STB, but how does that STB handle all this different information? We will describe now in more detail how product entitlement occurs in an STB.

Figure 2.10 depicts part of an STB. The conditional access (CA) Client receives information in the TS that allows it to calculate the key under which the video and audio streams are encrypted. This key is then passed to the system-on-chip (SoC) which decrypts the video content and renders it. The SoC and CA client are implemented separately and in most STBs the CA client is embedded in a tamper resistant smart card but it is also possible to do it in software. This approach allows the CA client to be easily changed without having to reproduce the video rendering hardware. We will now describe the exact process that an STB follows to decrypt a video broadcast.

The STB first receives a unique EMM. This is an EMM that is intended for a specific CA client, but since all STBs can read the EMM stream this EMM is encrypted under a master key (MK) which is unique to the CA client for which the EMM is intended. This EMM contains a selection of broadcast encryption (BE) keys. These keys are used with the broadcast encryption scheme that the broadcaster has chosen. Once the BE keys

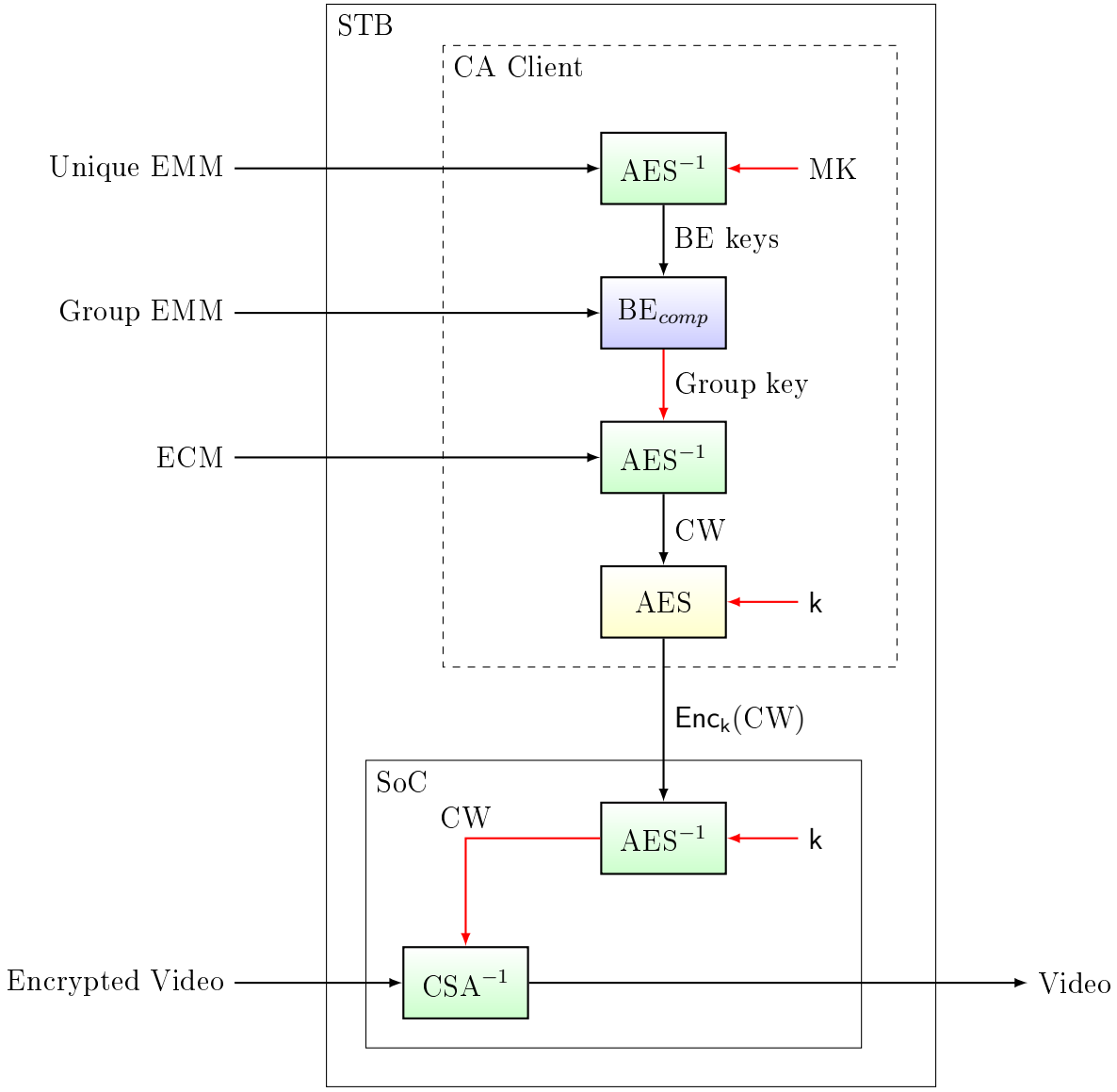


Figure 2.10: Inside an STB

have been decrypted the STB waits for the correct group EMM to be received. Typically each receiver is assigned to a group of 1024 STBs [21] due to restrictions on how large the EMMs can be. A group EMM is intended only to be used by receivers in that group. The contents of such a group EMM can only be utilised correctly using the BE keys sent to receivers in a unique EMM. The BE_{comp} steps takes the BE keys together with the group EMM and computes a new key called a group key. For all receivers in a group that are entitled to decrypt a broadcast this calculated group key will be the same.

Each ECM contains a single encrypted key called a code word (CW). This CW is essentially 48 bits long, which can be brute forced within a day [15]. For this reason the CW used to encrypt the video broadcast is changed every ten seconds as it is still infeasible to find a 48 bit key in such a short time. These CW are decrypted using the previously computed group key. This CW is then encrypted again under a key k, which is

unique to the STB's SoC, to protect it when being transferred from the CA client to the SoC. Once inside the SoC the CW is decrypted and then used as key input to the common scrambling algorithm (CSA) to decrypt the video content. The use of this algorithm is mandated by the DVB-CSA standard. Finally, the SoC renders the video and outputs it to the connected television set.

2.8 Summary

In this chapter we looked at the field of cryptography and some of the various primitives and security definitions it uses to construct secure schemes. An overview of white-box cryptography is given as we will design our scheme to be secure in the white-box attack model. Our particular focus falls more in the public key cryptography domain and then we described some modern broadcast encryption schemes. Finally we looked at how a set top box interacts with broadcasted information to extract key information to send to the conditional access client so we can design a product entitlement scheme that will work within this framework.

We have now laid the necessary groundwork to proceed to design a bandwidth efficient broadcast encryption scheme to be used in a product entitlement system.

Chapter 3

Broadcast Encryption Schemes

3.1 Introduction

In this chapter we will detail two existing broadcast encryption schemes and then several of our attempts at designing a secure broadcast encryption scheme. These broadcast encryption schemes will be used to implement a product entitlement system. The two main design considerations for our broadcast encryption scheme are:

Collusion resistance: the scheme must be secure against any number of colluders operating in the white-box attack model. We make no assumptions on the financial aid available to the colluders. Thus they can buy any number of legitimate decoders that they require and we have to design for this case.

Bandwidth efficient: we prioritize low bandwidth usage in the scheme. As the system is being designed to run on satellite and terrestrial broadcasting networks, bandwidth is at a premium compared to storage, memory and computing power. A scheme will be bandwidth efficient if the number of keys in the message header is independent of the size of the total population and entitled set.

All the schemes detailed here are non-dynamic schemes. This means that an upper limit on the total viewer population is set during the setup phase of the scheme. Thus the scheme will be defined by the following set of algorithms:

- $\mathsf{Setup}(\lambda, n)$: Takes as input a security parameter λ and a number of viewers n, and outputs the parameter sets $\mathcal{P}$ and $\mathcal{S}$. Each viewer i only receives their parameters $\mathcal{P}_i \in \mathcal{P}$ while the centre keeps the set of system parameters $\mathcal{S}$ secret.

- $\mathsf{Encrypt}(\mathbb{S}, \mathcal{S})$: Takes as input the set of entitled viewer indices $\mathbb{S} \subseteq \{1, \ldots, n\}$ and set of system parameters $\mathcal{S}$. It outputs a pair (D, K). Here D is information given to all receivers and is known as the message header. K is the key used to encrypt the message that is to be broadcasted.

- $\mathsf{Decrypt}(\mathbb{S}, i, \mathcal{P}_i, D)$: Takes as input a subset $\mathbb{S} \subseteq \{1, \ldots, n\}$, a viewer index $i \in$

$\{1, \ldots, n\}$, viewer parameters $\mathcal{P}_i$ for viewer i and message header D. If $i \in \mathbb{S}$, then the algorithm outputs the message encryption key K, otherwise a random value $K' \neq K$.

The system is required to be correct, meaning that for all $\mathbb{S} \subseteq \{1, \ldots, n\}$ and all $i \in \mathbb{S}$, $\mathsf{Decrypt}(\mathbb{S}, i, \mathcal{P}_i, D) = K$ if $\mathcal{P}, \mathcal{S} \leftarrow \mathbf{Setup}(\lambda, n)$ and $D, K \leftarrow \mathsf{Encrypt}(\mathbb{S}, \mathcal{S})$.

We warn the reader upfront that none of our proposed schemes achieve the first design requirement, collusion resistance. We will first detail the construction for each of the two existing schemes and then move on to our newly proposed schemes. It will also be shown how to break the security of each of the newly designed schemes. For one scheme we supply a security proof. We then show how this proof is flawed and how to efficiently break the system. As our attempts at designing a collusion resistant scheme were unsuccessful, methods to protect the calculated key from being extracted directly from memory were not investigated.

3.2 Existing Schemes

When designing any system it is important to compare it to existing systems in order to evaluate its effectiveness. We have identified bandwidth efficiency as the most important design consideration. In Table 3.1, taken from [41], a variety of broadcast encryption schemes are listed together with the theoretical amount of bandwidth each scheme will use. Here k indicates the maximum size of the collusion against which the scheme stays secure. From the this table it can clearly be seen that BGW_1 and DPP_1 have the best bandwidth efficiency of the listed schemes and we will thus compare our designed schemes practically against these schemes.

We will now describe how each of these two schemes works before moving on to describe our designed schemes.

3.3 BGW_1

3.3.1 Overview

The scheme first generates a sequence of values $g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}$. Note that the value g_{n+1} is missing from this sequence and the security assumption is that g_{n+1} cannot be found from this sequence alone. The scheme also defines a bilinear map $e(\cdot, \cdot)$ for the sequence and to gain access to the broadcast each viewer must calculate $e(g_{n+1}, g^t)$ when receiving g^t from the broadcast centre.

Table 3.1: List of Broadcast Encryption Schemes

Scheme	Bandwidth overhead	Storage at centre	Storage at receiver
ABBE [41]	$O(\log n)$	N/A	$O(\log n + m)$
Subset-Diff [16]	$O(k^2 \cdot \log^2 k \cdot \log n)$	$O(n)$	$O(k \cdot \log k \cdot \log n)$
BGW_1**[4]**	$O(1)$	N/A	$O(n)$
BGW_2[4]	$O(n^{\frac{1}{2}})$	N/A	$O(n^{\frac{1}{2}})$
NNL_1[30]	$O(k \log(n/k))$	N/A	$O(\log n)$
NNL_2[30]	$O(k)$	N/A	$O(\log^2 n)$
DPP_1**[12]**	$O(1)$	N/A	$O(n)$
DPP_2[12]	$O(k)$	N/A	$O(1)$
BW [7]	$O(n^{\frac{1}{2}})$	N/A	$O(n^{\frac{1}{2}})$
LT [38]	$O(k)$	N/A	$O(\log n)$
ACP [42]	$O(n)$	$O(n)$	$O(1)$
Flat-Table [9]	$O(\log n)$	$O(\log n)/O(n)$	$O(\log n)$
Flat-Table-ABE [9]	$O(\log^2 n)$	$O(\log n)/O(n)$	$O(\log n)$

3.3.2 Setup(λ, n):

On input (λ, n) generate a prime p with $\|p\| = \lambda$ and bilinear groups $\mathbb{G}, \mathbb{G}_1$ of order p with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$. Pick a random generator $g \leftarrow \mathbb{G}$ and integer $\alpha \leftarrow \mathbb{Z}_p$. Compute the elements $g_i = g^{\alpha^i}$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$. Note that g_{n+1} is not calculated. Pick a random integer $\gamma \leftarrow \mathbb{Z}_p$ that will be kept secret by the centre. The viewer parameters are

$$\mathcal{P}_i = (e(\cdot,\cdot), \langle g_i \rangle, g^{\gamma}, g_i^{\gamma}) \tag{3.1}$$

and the system parameters are

$$\mathcal{S} = (e(\cdot,\cdot), \langle g_i \rangle, g^{\gamma}) \tag{3.2}$$

where $\langle g_i \rangle = \left\{ g^{\alpha^i} | i = 1, 2, \ldots, n, n+2, \ldots, 2n \right\}$.

3.3.3 Encrypt$(\mathbb{S}, \mathcal{S})$:

On input $(\mathbb{S}, \mathcal{S})$ choose a random element $t \leftarrow \mathbb{Z}_p$, set the encryption key

$$K = e(g_n, g_1)^t = e(g^{\alpha^n}, g^{\alpha})^t = e(g, g)^{\alpha^{n+1}t} \tag{3.3}$$

and the message header

$$D = \left(g^t, (g^{\gamma} \cdot \prod_{j \in \mathbb{S}} g_{n+1-j})^t \right). \tag{3.4}$$

3.3.4 Decrypt$(\mathbb{S}, i, \mathcal{P}_i, D)$:

On this input, with message header $D = \left(g^t, (g^\gamma \cdot \prod_{j\in\mathbb{S}} g_{n+1-j})^t\right)$, viewer i calculates the decryption key

$$\begin{aligned}
& e\left(g_i, (g^\gamma \cdot \prod_{j\in\mathbb{S}} g_{n+1-j})^t\right) / e\left(g^t, \left(g_i^\gamma \cdot \prod_{j\in\mathbb{S}, j\neq i} g_{n+1-j+i}\right)\right) \\
= \; & e\left(g^{\alpha^i}, (g^\gamma \cdot \prod_{j\in\mathbb{S}} g_{n+1-j})^t\right) / e\left(g^t, \left(g^{\alpha^i\gamma} \cdot \prod_{j\in\mathbb{S}, j\neq i} g_{n+1-j+i}\right)\right), \text{ since } g_i = g^{\alpha^i} \\
= \; & e\left(g^{\alpha^i}, (g_{n+1-i})^t\right) \cdot e\left(g, (g^\gamma \cdot \prod_{j\in\mathbb{S}, j\neq i} g_{n+1-j})\right)^{\alpha^i t} / e\left(g, \left(g^{\alpha^i\gamma} \cdot \prod_{j\in\mathbb{S}, j\neq i} g_{n+1-j+i}\right)\right)^t \\
= \; & e\left(g, g^{\alpha^{n+1-i}t}\right)^{\alpha^i} \cdot e\left(g, (g^\gamma \cdot \prod_{j\in\mathbb{S}, j\neq i} g_{n+1-j})\right)^{\alpha^i t} / e\left(g, \left(g^\gamma \cdot \prod_{j\in\mathbb{S}, j\neq i} g_{n+1-j+i}\right)\right)^{\alpha^i t} \\
= \; & e\left(g, g\right)^{\alpha^i \alpha^{n+1-i} t} \\
= \; & e\left(g, g\right)^{\alpha^{n+1} t} \\
= \; & K
\end{aligned} \tag{3.5}$$

which is identical to the original encryption key used to encrypt the message using symmetric encryption.

3.3.5 Security

This construction bases its security on the assumption that the ℓ-Bilinear Diffie-Hellman Exponent assumption is hard. This assumption is defined as follows:

Definition 31. ℓ-Bilinear Diffie-Hellman Exponent Assumption

Let G be a bilinear group of prime order p with bilinear map $e : G \times G \to G_1$. Given the vector of $2\ell + 1$ elements

$$\left(h, g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^\ell}, g^{\alpha^{\ell+2}}, \ldots, g^{\alpha^{2\ell}}, \right) \in G^{2\ell+1}$$

calculate $e(g, h)^{\alpha^{\ell+1}}$.

The decryption key in the scheme adds an additional value t to the exponent to allow for multiple encryptions to take place.

3.4 DPP_1

3.4.1 Overview

In this scheme each viewer is assigned a linear polynomial of the form $\gamma + x_i$ with γ unknown and x_i known. While γ and x_i are constant for the scheme it helps to conceptuality regard γ as the variable in a linear polynomial. The viewer is given a value which contains $\frac{x_i}{\gamma+x_i}$ in the exponent and to calculate the session key they must remove the $\frac{1}{\gamma+x_i}$ part from the exponent. Because γ is kept secret from the viewer they cannot do so easily. The broadcast centre then includes information in the message header which the viewer can use to introduce $\frac{\gamma}{\gamma+x_i}$ into the exponent and then calculate $\frac{x_i}{\gamma+x_i} + \frac{\gamma}{\gamma+x_i} = 1$ and thus removing $\frac{1}{\gamma+x_i}$.

3.4.2 $\mathsf{Setup}(\lambda, n)$:

On input (λ, n) generate a prime p with $\|p\| = \lambda$ and bilinear groups $\mathbb{G}, \mathbb{G}_1$ of order p with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$. Choose two random generators $(g, h) \leftarrow \mathbb{G}$, and an integer $\gamma \leftarrow \mathbb{Z}_p$ that is kept secret by the centre and set $w = g^\gamma$. For each viewer i select an integer $x_i \leftarrow \mathbb{Z}_p$ and set $a_i = g^{\frac{x_i}{\gamma+x_i}}, b_i = h^{\frac{1}{\gamma+x_i}}$. The viewer parameters are

$$\mathcal{P}_i = (e(\cdot,\cdot), a_i, \langle x_i \rangle, \langle b_i \rangle) \tag{3.6}$$

and the system parameters are

$$\mathcal{S} = (e(\cdot,\cdot), g, h, w, \gamma, \langle x_i \rangle)\,. \tag{3.7}$$

3.4.3 $\mathsf{Encrypt}(\mathbb{S}, \mathcal{S})$:

On input $(\mathbb{S}, \mathcal{S})$ set $\mathbb{R} = \bar{\mathbb{S}}$ and choose a random $t \leftarrow \mathbb{Z}_p$. Calculate $r = \prod_{i\in\mathbb{R}} \frac{1}{\gamma+x_i}$, set message header

$$D = \left(w^t, h^{tr}\right) \tag{3.8}$$

and encryption key

$$K = e(g, h)^{tr}. \tag{3.9}$$

3.4.4 $\mathsf{Decrypt}(\mathbb{S}, i, \mathcal{P}_i, D)$:

On the input viewer i calculates the revoked set $\mathbb{R} = \bar{\mathbb{S}}$ and b_i^r with $r = \prod_{j\in\mathbb{R}} \frac{1}{\gamma+x_j}$. An algorithm which can be used to find b_i^r is described later. Next the viewer calculates the

decryption key

$$\begin{aligned} & e\left(w^t, b_i^r\right) \cdot e\left(a_i, h^{tr}\right) \\ = \; & e\left(g^\gamma, h^{\frac{1}{\gamma+x_i}}\right)^{tr} \cdot e\left(g^{\frac{x_i}{\gamma+x_i}}, h^{tr}\right) \\ = \; & e\left(g,h\right)^{\frac{\gamma tr}{\gamma+x_i}} \cdot e\left(g,h\right)^{\frac{x_i tr}{\gamma+x_i}} \\ = \; & e\left(g,h\right)^{\frac{(\gamma+x_i)tr}{\gamma+x_i}} \\ = \; & e(g,h)^{tr} \\ = \; & K \end{aligned} \tag{3.10}$$

which is the original encryption key.

In order to calculate this session key the viewer must know the value b_i^r with

$$r = \prod_{j \in \mathbb{R}} \frac{1}{\gamma + x_j}. \tag{3.11}$$

While every viewer knows the value of all x_i, without knowledge of γ they cannot straightforwardly calculate r. Delerablée et al. [12] describes a generic method that is used in all three schemes they present for calculating b_i^r. Because we only use one of the schemes they propose we will present the method we found and use to calculate b_i^r.

From the definition $b_i = h^{\frac{1}{\gamma+x_i}}$ it follows that

$$b_i^r = h^{\frac{1}{\gamma+x_i} \cdot \Pi_{j \in \mathbb{R}} \frac{1}{\gamma+x_j}} \tag{3.12}$$

and the exponent can be rewritten in its partial fraction expansion

$$\frac{1}{\gamma + x_i} \cdot \prod_{j \in \mathbb{R}} \frac{1}{\gamma + x_j} = \frac{\psi_i}{\gamma + x_i} + \sum_{j \in \mathbb{R}} \frac{\psi_j}{\gamma + x_j} \tag{3.13}$$

which implies that

$$\begin{aligned} b_i^r &= h^{(\frac{\psi_i}{\gamma+x_i} + \sum_{j \in \mathbb{R}} \frac{\psi_j}{\gamma+x_j})} \\ &= h^{\frac{\psi_i}{\gamma+x_i}} \cdot \prod_{j \in \mathbb{R}} h^{\frac{\psi_j}{\gamma+x_j}} \\ &= b_i^{\psi_i} \cdot \prod_{j \in \mathbb{R}} b_j^{\psi_j}. \end{aligned} \tag{3.14}$$

If a viewer solves for all ψ they can calculate b_i^r without knowledge of γ because all viewers know all values of x_i and b_i.

3.4.5 Security

The security of this construction is based on the assumption that the General Diffie-Hellman Exponent and General Decisional Diffie-Hellman Exponent problems are computationally infeasible. These problems are defined as [12]:

Definition 32. General Diffie-Hellman Exponent Problem

Let $\mathbb{G}, \mathbb{G}_1$ be cyclic groups of order p with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$. Set g_0 a generator of $\mathbb{G}$ and $g = e(g_0, g_0)$. If P and Q are two s-tuple of m-variate polynomials with $P = (1, p_2, \ldots, p_s)$ and Q similarly we define F as

$$F = \sum_{1 \leq i,j \leq s} a_{i,j} p_i p_j + \sum_{1 \leq i \leq s} b_i q_i$$

with $a_{i,j}, b_i \in \mathbb{Z}_p$.

Given the vector

$$H(x_1, \ldots, x_m) = (g_0^{p_1(x_1,\ldots,x_m)}, \ldots, g_0^{p_s(x_1,\ldots,x_m)}, g^{q_1(x_1,\ldots,x_m)}, \ldots, g^{q_s(x_1,\ldots,x_m)}) \in G^s \times G_T^s,$$

compute $g^{F(x_1,\ldots,x_m)}$.

Definition 33. General Decisional Diffie-Hellman Exponent Problem

Given the vector $H(x_1, \ldots, x_m) \in G^s \times G_T^s$ as above and $T \in G_T$, decide if $T = g^{F(x_1,\ldots,x_m)}$.

3.5 The Initial Design: Hiding the Group Order

3.5.1 Design Choices

The first design tried to utilize subgroups in elliptic curves of non-prime order. Such elliptic curve groups were used by Boneh et al. [5] to construct a homomorphic encryption scheme which allows for infinite additions but only one multiplication to take place. While the entire group order, a semiprime, is known, the scheme operates in a subgroup that is of order one of the prime factors of the total group order. This has the effect of creating a hidden group order that is unknown to the attacker. It has to be noted that this group order is only hidden so long as the attacker is unable to factor the total group order. Thus the sizes of the primes must be chosen accordingly. In our scheme if an attacker executes the $\mathsf{Decrypt}(\mathbb{S}, i, \mathcal{P}_i, D)$ algorithm they would calculate a value K^{a_i}, where K is the session key and the assumption was made that an attacker would not be able to remove a_i from the exponent without knowing the hidden group order. This assumption is shown to be false in Section 3.5.6.

3.5.2 Overview

As mentioned before the order of the group used in the scheme is a semiprime. From this group two subgroups are created with distinct orders, each of which has a unique generator. One of these generators is used to create a blinding factor and the other to calculate the session key. Each viewer is assigned a user parameter a_i and all viewers in the system know all a_i. Due to this the blinding factor is added in an attempt to mitigate any information leakage due to the knowledge of all the a_i. The generators of the subgroups are chosen in such a way that the blinding factor is removed during the pairing operation of a bilinear map $e(\cdot,\cdot)$. The message header contains the inverses of all the a_i values of the viewers in the entitled set and viewers can remove these inverses by exponentiating with the correct a_i values.

3.5.3 Setup(λ, n):

On input (λ, n) generate two primes p and q with $\|p\| = \|q\| = \frac{\lambda}{2}$ and then generate the elliptic curve groups $\mathbb{G}$ and $\mathbb{G}_T$ of order $N = pq$ with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let g be a generator of $\mathbb{G}$ so that $e(g,g)$ is a generator of $\mathbb{G}_T$ and from Theorem 18 we have that $e\left(g,g\right)^N = 1$. The scheme requires a generator g_p of a subgroup of order p and g_q a generator of a subgroup of order q. These generators are found by setting $g_p = g^q$ and $g_q = g^p$. Choose a value $\gamma \leftarrow \mathbb{Z}_p$ that is kept secret by the centre.

For each viewer i in the viewer population choose a unique viewer exponent $a_i \leftarrow \mathbb{Z}_p$ and blinding value exponent $r_i \leftarrow \mathbb{Z}_q$. The a_i values can be chosen using a pseudorandom function $F(\cdot)$ and setting $a_i = F(i) \bmod p$. The user parameters are

$$\mathcal{P}_i = \left(N, e(\cdot,\cdot), F(\cdot), g, g_q^{r_i} g_p^{a_i \gamma}\right) \tag{3.15}$$

where $e(\cdot,\cdot)$ and $F(\cdot)$ respectively refer to the definitions of the bilinear map and pseudo-random functions. The system parameters are

$$\mathcal{S} = \left(p, q, F(\cdot), g, e(g_p, g_p)^{\gamma}\right). \tag{3.16}$$

3.5.4 Encrypt$(\mathbb{S}, \mathcal{S})$:

Let $B_{\mathbb{S}} = \prod_{i \in \mathbb{S}} a_i^{-1}$. For each broadcast select a fresh $t \leftarrow \mathbb{Z}_p$ and set the message header

$$D = g_p^{tB_{\mathbb{S}}} \tag{3.17}$$

and the encryption key

$$K = e(g_p, g_p)^{\gamma t}. \tag{3.18}$$

3.5.5 Decrypt$(\mathbb{S}, i, \mathcal{P}_i, D)$:

On the input the viewer i first calculates

$$\begin{aligned} e(g_q^{r_i} g_p^{a_i\gamma}, D) &= e(g_q^{r_i} g_p^{a_i\gamma}, g_p^{tB_{\mathbb{S}}}) \\ &= e(g_q^{r_i}, g_p^{tB_{\mathbb{S}}}) \cdot e(g_p^{a_i\gamma}, g_p^{tB_{\mathbb{S}}}) \\ &= e(g^p, g^q)^{r_i dB_{\mathbb{S}}} \cdot e(g_p, g_p)^{a_i\gamma tB_{\mathbb{S}}} \\ &= \left(e(g,g)^N\right)^{r_i tB_{\mathbb{S}}} \cdot e(g_p, g_p)^{a_i\gamma tB_{\mathbb{S}}} \\ &= e(g_p, g_p)^{\gamma tB_{\mathbb{S}} a_i} \end{aligned} \tag{3.19}$$

and the viewer then calculates the value $\hat{A}_{\mathbb{S}} = \prod_{j\in\mathbb{S}, i\neq j} a_j$. Note that $a_i \hat{A}_{\mathbb{S}} B_{\mathbb{S}} = 1$. Finally the viewer calculates the decryption key as

$$\left(e(g_p, g_p)^{\gamma tB_{\mathbb{S}} a_i}\right)^{\hat{A}_{\mathbb{S}}} = e(g_p, g_p)^{\gamma t\left(B_{\mathbb{S}} a_i \hat{A}_{\mathbb{S}}\right)} = e(g_p, g_p)^{\gamma t} = K \tag{3.20}$$

which is the required session key.

3.5.6 Cryptanalysis

There are two easy ways to attack the above scheme. The first is with the aid of a single colluder. If both attackers execute Decrypt$(\mathbb{S}, i, \mathcal{P}_i, D)$ they will calculate two values K^{a_i} and K^{a_j} with i and j distinct and thus a_i and a_j distinct. If $\gcd(a_i, a_j) = 1$, which for randomly chosen a_i, a_j does not happen with negligible probability [31], the attackers can find integers X, Y such that $Xa_i + Ya_j = 1$. They can then calculate the value $(K^{a_i})^X (K^{a_j})^Y = K^{Xa_i + Ya_j} = K$ which is the required session key and the scheme is broken. This attack is known as the textbook RSA attack [20]. If all the a_i are chosen such that they all share a common factor $\omega > 1$ and thus $\gcd(a_i, a_j) = \omega$ the integers X and Y cannot be found, but this does not make the scheme secure. Since $\gcd(a_i, a_j) = \omega$ we can write

$$a_i = \omega v_i, a_j = \omega v_j \tag{3.21}$$

with v_i and v_j relatively prime. If the entitled set consists only of a single viewer z then

$$D = g^{ta_z^{-1}} = g^{t\omega^{-1} v_z^{-1}} \tag{3.22}$$

and instead of multiplying by a_z when calculating $\hat{A}_{\mathbb{S}}$ the attackers multiply by v_z. This will result in the value obtained at the end of executing Decrypt$(\mathbb{S}, i, \mathcal{P}_i, D)$ to have ω^{-1} in the exponent:

$$K^{\omega^{-1} a_i} = K^{v_i}. \tag{3.23}$$

Similarly K^{v_j} can be found and since $\gcd(v_i, v_j) = 1$ the textbook RSA attack can still be applied.

The second way of attacking this scheme requires no additional colluders and shows that the inverse of any a_i can be calculated, even without knowing the order of the subgroups. It is done by using the following lemma:

Lemma 34. *For any $a \in \mathbb{Z}_p$ for which there exists an inverse b in $\mathbb{Z}_{kp}$ for some $k \in \mathbb{Z}^+$, b also acts as an inverse in $\mathbb{Z}_p$.*

Proof. Since b is an inverse in $\mathbb{Z}_{kp}$ it holds that $ab = \alpha kp + 1$ for some $\alpha \in \mathbb{Z}$. Therefore $ab = \alpha kp + 1 = 1 \bmod p$. Thus b also acts as an inverse for a in $\mathbb{Z}_p$. □

Since the group order N is known to the attacker they can simply calculate a value b_i which acts as an inverse for a_i modulo N and use the same value as an inverse modulo p since $N = pq$. The only case where a_i does not have in inverse in $\mathbb{Z}_N$ is when $\gcd(a_i, N) \neq 1$. But since N only has two prime factors this means the gcd is either p or q and thus the attacker can factor N by calculating this gcd.

3.6 The Rescue Attempt: True Hidden Group Orders

3.6.1 Design Choices

There are two main security flaws in the previous design: the textbook RSA attack derivative and that the attacker can calculate the inverse of any exponent they wish. To address the latter the scheme was moved over to the group $\mathbb{Z}_N^*$ on which RSA is based. This group would not allow an attacker to calculate an inverse of an exponent. Unfortunately the scheme can no longer use bilinear maps and a different method of removing the blinding factor had to be used.

Fiat and Naor proposed a similar scheme in [16] but in their scheme there is no blinding factor present.

3.6.2 Overview

This scheme is similar to the one previously presented. Each viewer is assigned an exponent a_i that is known by all other viewers. The inverses of the exponents of viewers in the entitled group is added to the exponent of the message header and a viewer removes them by exponentiating with the correct a_i values. Since we can no longer define a bilinear map to be used on the keys the blinding factor is removed by exponentiation.

3.6.3 Setup(λ, n):

On input (λ, n) generate two primes p and q with $\|p\| = \|q\| = \frac{\lambda}{2}$ and values $\gamma \leftarrow \mathbb{Z}^*_{\phi(N)}$, $m \leftarrow \mathbb{Z}^*_N$. For each viewer i choose the viewer exponent $a_i \leftarrow \mathbb{Z}^*_{\phi(N)}$, blinding base $h_i \leftarrow \mathbb{Z}^*_N$, blinding value $e_i \leftarrow \mathbb{Z}^*_N \setminus \mathbb{Z}^*_{\phi(N)}$ and calculate the blinding exponent $d_i = \frac{\phi(N)}{e_i}$. The e_i must be chosen so that $\phi(N) \not| \prod_{\forall i} e_i$. The values a_i can again be chosen with some pseudorandom function $F(\cdot)$. The viewer parameters $\mathcal{P}_i$ is set to $\left(N, F(\cdot), h_i^{d_i} m^{\gamma a_i}\right)$. The system parameters $\mathcal{S}$ is set to $\left(p, q, m^\gamma, \langle e_i \rangle, F(\cdot)\right)$.

3.6.4 Encrypt$(\mathbb{S}, \mathcal{S})$:

On input $(\mathbb{S}, \mathcal{S})$ choose $t \leftarrow \mathbb{Z}^*_{\phi(N)}$ and calculate $E_\mathbb{S} = \prod_{i \in \mathbb{S}} e_i$ and $B_\mathbb{S} = \prod_{i \in \mathbb{S}} a_i^{-1}$. Set message header

$$D = t B_\mathbb{S} E_\mathbb{S} \pmod N \tag{3.24}$$

and encryption key

$$K = m^{E_\mathbb{S} \gamma t} \pmod N. \tag{3.25}$$

Note that for any $i \in \mathbb{S}$, $d_i E_\mathbb{S} = \frac{\phi(N) E_\mathbb{S}}{e_i} = 0 \bmod \phi(N)$.

3.6.5 Decrypt$(\mathbb{S}, i, \mathcal{P}_i, D)$:

When receiving the description of the entitled set $\mathbb{S}$, viewer i calculates $\hat{A}_\mathbb{S} = \prod_{j \in \mathbb{S}, i \neq j} a_j$ (again $a_i \hat{A}_\mathbb{S} B_\mathbb{S} = 1$) and then

$$\begin{aligned}
\left(h_i^{d_i} m^{\gamma a_i}\right)^{\hat{A}_\mathbb{S} D} &= \left(h_i^{d_i} m^{\gamma a_i}\right)^{\hat{A}_\mathbb{S} t B_\mathbb{S} E_\mathbb{S}} \pmod N \\
&= \left(h_i^{d_i a_i^{-1}} m^{\gamma}\right)^{a_i \hat{A}_\mathbb{S} B_\mathbb{S} E_\mathbb{S} t} \pmod N \\
&= \left(h_i^{a_i^{-1}}\right)^{d_i E_\mathbb{S} t} (m^\gamma)^{E_\mathbb{S} t} \pmod N \\
&= m^{E_\mathbb{S} \gamma t} \pmod N \\
&= K.
\end{aligned} \tag{3.26}$$

3.6.6 Cryptanalysis

3.6.6.1 Textbook RSA Attack with Two Colluders

To illustrate the collusion resistance, take two colluding viewers x and y and two entitled users v and w. The centre broadcasts the key $k_d \left(a_v^{-1} e_v\right) \left(a_w^{-1} e_w\right)$ and any viewer j not in

$\mathbb{S}$ calculates:

$$\begin{aligned}\left(h_j^{d_j} m^{\gamma a_j}\right)^{\left[t\left(a_v^{-1}e_v\right)\left(a_w^{-1}e_w\right)\right]a_v a_w} &= \left(h_j^{d_j} m^{\gamma a_j}\right)^{te_v e_w a_v^{-1} a_v a_w^{-1} a_w} \\ &= \left(h_j^{d_j} m^{\gamma a_j}\right)^{te_v e_w} \\ &= \left(h_j^{d_j}\right)^{te_v e_w} \left(m^{k_s a_j}\right)^{te_v e_w} \\ &= \left(h_j^{tE_{\mathbb{S}}}\right)^{d_j} \left(m^{E_{\mathbb{S}}\gamma t}\right)^{a_j} \\ &= \left(h_j^{tE_{\mathbb{S}}}\right)^{d_j} (K)^{a_j}. \end{aligned} \tag{3.27}$$

Now assume the collusion of x and y calculates values X and Y such that $Xa_x + Ya_y = 1$. The collusion now attempts to apply the textbook RSA attack:

$$\begin{aligned}&\left(\left(h_x^{tE_{\mathbb{S}}}\right)^{d_x} (K)^{a_x}\right)^X \cdot \left(\left(h_y^{tE_{\mathbb{S}}}\right)^{d_y} (K)^{a_y}\right)^Y \\ &= \left(\left(h_x^{tE_{\mathbb{S}}}\right)^{Xd_x} \cdot \left(h_y^{tE_{\mathbb{S}}}\right)^{Yd_y}\right) \left((K)^{Xa_x + Ya_y}\right) \\ &= \left(\left(h_x^{Xd_x}\right)^{tE_{\mathbb{S}}} \cdot \left(h_y^{Yd_y}\right)^{tE_{\mathbb{S}}}\right) \cdot \left((K)^1\right) \\ &= \left(h_x^{Xd_x E_{\mathbb{S}}} \cdot h_y^{Yd_y E_{\mathbb{S}}}\right)^t \cdot K. \end{aligned} \tag{3.28}$$

While the attack might have isolated the session key K as a factor, the value calculated still contains a blinding factor of the form $\left(\left(h_x^{Xd_x E_{\mathbb{S}}}\right)\left(h_y^{Yd_y E_{\mathbb{S}}}\right)\right)^t$.

The derived value in (3.28) is still protected by a blinding factor. But this factor only remains a blinding factor as long as $Xd_x E_{\mathbb{S}} \neq 0 \bmod \phi(N)$ and $Yd_y E_{\mathbb{S}} \neq 0 \bmod \phi(N)$. If both these conditions are true the collusion of two viewers can successfully calculate the session key K. Since there is a limited number of choices for each e_i, the probability that $d_x E_{\mathbb{S}} = 0 \bmod \phi(N)$ is much higher than $XE_{\mathbb{S}} = 0 \bmod \phi(N)$ and similarly for Y, we will focus on the former.

Let $\phi(N)$ have $\omega + 1$ distinct prime factors, denoted as $\mathbb{E}$, which results in ω choices for each e_i. Let $\mathbb{K}$ be the set of colluders with $|\mathbb{K}| = k$ and $|\mathbb{S}| = s$. We assume that the set of keys held by the colluders is chosen uniformly at random. This is because the collusion has no control over the assignment of keys which is done uniformly at random. Let $\mathbb{E}_{\mathbb{S}} = \{e_i | i \in \mathbb{S}\}$. Thus $E_{\mathbb{S}} = \prod_{\forall e \in \mathbb{E}_{\mathbb{S}}} e$.

Assume now that $k = 2$ and that e_x and e_y is associated with the two colluders. The collusion breaks the security of the system if $e_x \in \mathbb{E}_{\mathbb{S}}$ and $e_y \in \mathbb{E}_{\mathbb{S}}$. We are only interested in the case where none of the colluders are in the entitled set i.e. $x \notin \mathbb{S}$ and $y \notin \mathbb{S}$. Thus we want to determine the probability

$$\Pr\left[e_x \in \mathbb{E}_{\mathbb{S}}, e_y \in \mathbb{E}_{\mathbb{S}}\right] \tag{3.29}$$

where the elements of $\mathbb{E}_{\mathbb{S}}$ is chosen uniformly at random from $\mathbb{E}$.

Since the choices of e_x, e_y and $\mathbb{E}_{\mathbb{S}}$ are uniformly at random and independent from each other the events $e_x \in \mathbb{E}_{\mathbb{S}}$ and $e_y \in \mathbb{E}_{\mathbb{S}}$ are statistically independent. It follows then that

$$\Pr\left[e_x \in \mathbb{E}_{\mathbb{S}}, e_y \in \mathbb{E}_{\mathbb{S}}\right] = \Pr\left[e_x \in \mathbb{E}_{\mathbb{S}}\right] \cdot \Pr\left[e_y \in \mathbb{E}_{\mathbb{S}}\right] = \Pr\left[e_x \in \mathbb{E}_{\mathbb{S}}\right]^2. \tag{3.30}$$

To determine $\Pr\left[e_x \in \mathbb{E}_{\mathbb{S}}\right]$ we note that this is the probability that a set of s elements chosen at random with replacement from a set of ω elements contains a specific element. The number of ways of choosing a set of s elements with replacement from a set of ω elements is given by $\binom{s+\omega-1}{s}$. The number of sets of size s that contain a specific element is the same as fixing the first choice to the element and then choosing the remaining $s-1$ elements with replacement. This number is given by $\binom{s+\omega-2}{s-1}$. Thus we have

$$\begin{aligned}
\Pr\left[e_x \in \mathbb{E}_{\mathbb{S}}\right] &= \frac{\binom{s+\omega-2}{s-1}}{\binom{s+\omega-1}{s}} \\
&= \frac{\frac{(s+\omega-2)!}{(s-1)!(\omega-1)!}}{\frac{(s+\omega-1)!}{s!(\omega-1)!}} \\
&= \frac{(s+\omega-2)!}{(s+\omega-1)!} \cdot \frac{s!}{(s-1)!} \cdot \frac{(\omega-1)!}{(\omega-1)!} \\
&= \frac{s}{s+\omega-1}.
\end{aligned} \tag{3.31}$$

Using this result we can now calculate the probability of success that a collusion of two viewers will have when applying the attack described in (3.28) as

$$\Pr\left[e_x \in \mathbb{E}_{\mathbb{S}}, e_y \in \mathbb{E}_{\mathbb{S}}\right] = \Pr\left[e_x \in \mathbb{E}_{\mathbb{S}}\right]^2 = \left(\frac{s}{s+\omega-1}\right)^2. \tag{3.32}$$

3.6.6.2 Textbook RSA Attack with $k > 2$ Colluders

For collusions with a size larger than two, any two users in the collusion can try to apply the attack. The probability to break the scheme is therefore the probability that any subset of two colluders exist that satisfy $e_x \in \mathbb{E}_{\mathbb{S}}$ and $e_y \in \mathbb{E}_{\mathbb{S}}$.

The probability of this event not occurring is then the probability that no such subset satisfies $e_x \in \mathbb{E}_{\mathbb{S}}$ and $e_y \in \mathbb{E}_{\mathbb{S}}$. This probability can be described as

$$\Pr\left[e_x \notin \mathbb{E}_{\mathbb{S}} \wedge e_y \notin \mathbb{E}_{\mathbb{S}} | x \neq y \forall x, y \in \mathbb{K}\right]. \tag{3.33}$$

Since there are $\binom{k}{2}$ such subsets the probability can be calculated as

$$\begin{aligned}
&\Pr\left[e_x \notin \mathbb{E}_{\mathbb{S}} \wedge e_y \notin \mathbb{E}_{\mathbb{S}} | x \neq y \forall x, y \in \mathbb{K}\right] \\
&= \prod_{\forall x,y \in \mathbb{K}, x \neq y} \Pr\left[e_x \notin \mathbb{E}_{\mathbb{S}} \wedge e_y \notin \mathbb{E}_{\mathbb{S}}\right] \\
&= \prod_{\forall x,y \in \mathbb{K}, x \neq y} \left(1 - \Pr\left[e_x \in \mathbb{E}_{\mathbb{S}}, e_y \in \mathbb{E}_{\mathbb{S}}\right]\right) \\
&= \prod_{\forall x,y \in \mathbb{K}, x \neq y} \left(1 - \left[\frac{s}{s+\omega-1}\right]^2\right) \\
&= \left(1 - \left[\frac{s}{s+\omega-1}\right]^2\right)^{\binom{k}{2}} \\
&= \left(1 - \left[\frac{s}{s+\omega-1}\right]^2\right)^{\frac{k(k-1)}{2}}, \qquad (3.34)
\end{aligned}$$

finally giving the probability of breaking the scheme with a set of k colluders as

$$\Pr\left[e_x \in \mathbb{E}_{\mathbb{S}}, e_y \in \mathbb{E}_{\mathbb{S}}, \forall x, y \in \mathbb{K}, x \neq y\right] = 1 - \left(1 - \left[\frac{s}{s+\omega-1}\right]^2\right)^{\frac{k(k-1)}{2}}. \qquad (3.35)$$

Full collusion resistance for large viewer populations cannot be achieved due to the limited number of prime factors of $\phi(N)$. This value can be increased by making the modulus the product of more than two large prime numbers, but the larger the modulus the larger the message header will be. This will increase the value of ω in (3.35), thus reducing the overall probability.

3.6.6.3 Textbook RSA Attack with Multiple Moduli

But what if more than one modulus is used and several message headers are broadcast? Viewers could then be assigned different e_i and keys for each modulus and calculate a session key for each moduli. The different session keys are then combined in some way to form a single session key that is used to decrypt the content. To break this new scheme the collusion would have to break the scheme for each modulus used. The probability for calculating the correct key under a single modulus does not change. Thus if the new scheme uses μ moduli the probability of the collusion breaking the scheme is

$$\Pr\left[e_x \in \mathbb{E}_{\mathbb{S}}, e_y \in \mathbb{E}_{\mathbb{S}}, \forall x, y \in \mathbb{K} x \neq y\right]^{\mu} = \left(1 - \left[\frac{s}{s+\omega-1}\right]^2\right)^{\mu \cdot \frac{k(k-1)}{2}}. \qquad (3.36)$$

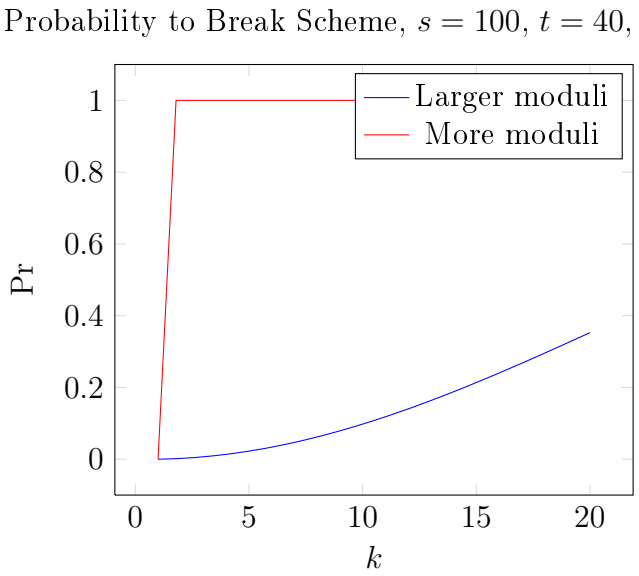


Figure 3.1: Comparison of larger moduli and more moduli strategies

If instead a modulus that has $\mu\omega$ prime factors available for use as e_i values and is μ times larger, the probability becomes

$$1 - \left(1 - \left[\frac{s}{s + \mu\omega - 1}\right]^2\right)^{\frac{k(k-1)}{2}}. \tag{3.37}$$

As can be seen from the graphs in Figure 3.1 for various parameters it is more beneficial to increase the size of the moduli (and thus ω) than broadcasting several moduli. This is with random assignment of each e_i to a viewer.

Full collusion resistance cannot be achieved even with multiple moduli and non-random assignment of keys. Full collusion resistance implies that if a single viewer is entitled and all others collude the collusion still cannot break the scheme. Recall that the scheme is broken if the collusion contains two viewers with the same e_i assigned to them as an e_i used as part of the message header. Thus if for any modulus the single entitled user has the same e_i as two or more other viewers the collusion can break the scheme for that particular modulus as the other viewers are part of the collusion. This means that at most each e_i can be shared between two users for each modulus and thus full collusion resistance can possible only be achieved for 2ω viewers.

3.7 A Rogue Attempt: Summing Elements from $\mathbb{Z}_N^*$

3.7.1 Design Choices

Again our designed scheme has been shown vulnerable to the textbook RSA attack. The blinding factors used up to now have had no effect in stopping the application of

this attack. Even when the blinding factors are still present in the calculated value the attackers can factor out the session key as shown in (3.28). This is due to the multiplicative nature of the blinding factors with the session key. The next design tried to remove this multiplicative property of the blinding factors by introducing a step where two parts of the key have to be summed for the blinding factor to be removed. Summing of integers in the set $\mathbb{Z}_N^*$ is not common practice as it no longer behaves as a group under addition modulo N, but in [32] Pointcheval introduces a scheme which does add 1 to an element chosen from $\mathbb{Z}_N^*$. They define two new problems relating to RSA: the Computational Dependent-RSA (C-DRSA) problem and the Extraction Dependent RSA (E-DRSA) problem:

Definition 35. The Computational Dependent-RSA (C-DRSA)

Given a large composite RSA modulus N, an integer e relatively prime to $\phi(N)$ and an integer $\alpha \in \mathbb{Z}_N^*$, find $(a+1)^e \bmod N$ where $\alpha = a^e \bmod N$.

and

Definition 36. The Extraction Dependent-RSA problem (E-DRSA)

Given a large composite RSA modulus N, an integer e relatively prime to $\phi(N)$, an integer $\alpha = a^e \bmod N$ and an integer $\gamma = (a+1)^e \bmod N$, find $a \bmod N$.

Pointcheval then proves that solving these two problems is equivalent to solving the RSA problem.

We aim to show that with high probability the set $\mathbb{Z}_N^*$ behaves as a group under addition modulo N for any randomly chosen pair of integers from the group and that this action does not introduce any security concerns. This is done by proving the computational hardness of factoring N and finding such a pair of integers is equivalent.

Lemma 37. *Pick two primes p, q with $\|p\| = \|q\| = \frac{\lambda}{2}$ and let $N = pq$. Then finding integers $x, y \in \mathbb{Z}_N^*$ with $x+y \neq N$ such that $(x+y) \bmod N \notin \mathbb{Z}_N^*$ is equivalent to factoring N.*

Proof. If $x + y \notin \mathbb{Z}_N^*$ it implies that $\gcd(x+y, N) \neq 1$. Because $N = pq$ with p and q both prime it holds that $\gcd(x+y, N) \in \{1, p, q, N\}, \forall x + y \in \mathbb{R}^+$. Since $x + y \neq N$ and $\gcd(x+y, N) \neq 1$ it holds that $\gcd(x+y, N) \in \{p, q\}$. Since $\gcd(x+y, N)$ is efficiently computable and is one of the prime factors of N, N can be factored.

For the reverse where p and q is known, choose $x \leftarrow \mathbb{Z}_N^*, r \leftarrow \mathbb{Z}_p$ and set $y = (qr - x) \bmod N$. □

This implies that if factoring is hard then so is finding two integers in $\mathbb{Z}_N^*$ whose sum modulo N is not in $\mathbb{Z}_N^*$. Since anyone can efficiently sample $\mathbb{Z}_N^*$ when knowing N it holds that summing random elements from $\mathbb{Z}_N^*$ will only reveal the factorisation of N with negligible probability. We use this result to construct a broadcast encryption scheme.

3.7.2 Overview

In this scheme there are two keys which are derived in the same manner. At the end these keys are summed together to produce the final decryption key. For each viewer the sum of these keys is constant but the two values that are summed is different for each viewer. If these keys are derived by a viewer not in the entitled set they will have an additional known exponent a_i. Unlike previous schemes these keys will not have the same base for different viewers and thus the textbook RSA attack will not work. At some point during the key calculation the key will contain a factor of the form $X_i^{b_i t}$ which needs to be removed prior to the summing of the elements; otherwise the sum will not be constant for any value of t. These factors are removed by supplying each viewer with a vector containing α values of the form $X_i^{r_i}$. These values are then used to evaluate a linear relationship in the exponent, similar to the approaches proposed by Matsumoto in [24], which is described in the message header and calculates the value $\left(X_i^{b_i}\right)^{1-t}$ which is used to remove the $X_i^{b_i t}$ factor from the key.

3.7.3 Setup(λ, n):

On input generate two primes p and q with $\|p\| = \|q\| = \frac{\lambda}{2}$ and let $N = pq$. Choose base $m \leftarrow \mathbb{Z}_N^*$, sum $C \leftarrow \mathbb{Z}_N^*$, secret value $\gamma \leftarrow \mathbb{Z}_{\phi(N)}^*$, vector size $\alpha \geq 2$ and vector exponents $r_j \leftarrow \mathbb{Z}_{\phi(N)}^*$ for $j = 0, \ldots, \alpha$.

For each viewer i choose a viewer exponent $a_i \leftarrow \mathbb{Z}_{\phi(N)}^*$, sum key $X_i \leftarrow \mathbb{Z}_N^*$ and let $b_i = a_i^{-1} \bmod \phi(N)$. Next let

$$\begin{aligned} Y_i &= \left(C - X_i^{b_i}\right)^{a_i} \pmod N \\ \Leftrightarrow C &= X_i^{b_i} + Y_i^{b_i} \pmod N. \end{aligned} \tag{3.38}$$

Calculate the vectors $\underline{X}^{(i)} = \left\{X_i^{b_i r_j} \,|j = 0, \ldots, \alpha\right\}$ and $\underline{Y}^{(i)} = \left\{Y_i^{b_i r_j} \,|j = 0, \ldots, \alpha\right\}$. The viewer parameters are

$$\mathcal{P}_i = \left(N, \langle a_i \rangle, X_i m^{\gamma a_i}, Y_i m^{\gamma a_i}, \underline{X}^{(i)}, \underline{Y}^{(i)}\right). \tag{3.39}$$

The system parameters are

$$\mathcal{S} = \left(\phi(N), \langle r_j \rangle, \langle b_i \rangle, m^{\gamma}, C\right). \tag{3.40}$$

3.7.4 Encrypt$(\mathbb{S}, \mathcal{S})$:

The centre chooses $t \leftarrow \mathbb{Z}^*_{\phi(N)}$ and calculates the vector $\underline{R_t} = \{R_x \,|x = 0, \ldots \alpha\}$ such that $t = 1 - \sum_j R_j r_j$. Let $B_{\mathbb{S}} = \prod_{i \in \mathbb{S}} b_i$. The centre then sets the encryption key

$$K = m^{\gamma t} C \tag{3.41}$$

and the message header

$$D = (tB_{\mathbb{S}}, \underline{R_t}) \,. \tag{3.42}$$

3.7.5 Decrypt$(\mathbb{S}, i, \mathcal{P}_i, D)$:

Viewer i first calculates $\prod_{j \in \mathbb{S}, i \neq j} a_i t B_{\mathbb{S}} = b_i t$. They then calculate $(X_i m^{\gamma a_i})^{b_i t} = X_i^{b_i t} m^{\gamma t}$. Similarly they find the value $(Y_i m^{\gamma a_i})^{b_i t} = Y_i^{b_i t} m^{\gamma t}$.

Using $\underline{R_t} = \left\{ R_x \middle| x = 0, \ldots \alpha; t = 1 - \sum_j R_j r_j \right\}$ and $\underline{X^{(i)}} = \left\{ X_i^{b_i r_j} \,|j = 0, \ldots, \alpha \right\}$ the viewer can calculate $\left(X_i^{b_i}\right)^{\sum_j R_j r_j} = \left(X_i^{b_i}\right)^{1-t}$. Similarly they can find $\left(Y_i^{b_i}\right)^{1-t}$ using $\underline{Y^{(i)}}$. Multiplying the newly calculated values with the previous results gives

$$\left(X_i^{b_i t} m^{\gamma t}\right) \left(X_i^{b_i}\right)^{1-t} = m^{\gamma t} \left(X_i^{b_i (t+1-t)}\right) = m^{\gamma t} X_i^{b_i} \tag{3.43}$$

and similarly $m^{\gamma t} Y_i^{b_i}$. Summing these values finally gives

$$m^{\gamma t} X_i^{b_i} + m^{\gamma t} Y_i^{b_i} = m^{\gamma t} \left(X_i^{b_i} + Y_i^{b_i}\right) = m^{\gamma t} C = K. \tag{3.44}$$

3.7.6 Cryptanalysis

This scheme unfortunately becomes entirely insecure once two viewers have been part of the entitled group only once. To see this note that

$$\begin{aligned} \left[m^{\gamma t} C\right]^{a_i} &= \left[m^{\gamma t} \left(Y_i^{b_i} + X_i^{b_i}\right)\right]^{a_i} \\ &= \left[m^{\gamma t} \left(Y_i^{b_i} \left[1 + \left(\frac{X_i}{Y_i}\right)^{b_i}\right]\right)\right]^{a_i} \\ &= \left(m^{\gamma t} Y_i^{b_i}\right)^{a_i} \left[1 + \left(\frac{X_i}{Y_i}\right)^{b_i}\right]^{a_i} . \end{aligned} \tag{3.45}$$

An attacker can calculate $\left(m^{\gamma t} Y_i^{b_i}\right)^{a_i}$ by executing the Decrypt$(\mathbb{S}, i, \mathcal{P}_i, D)$ algorithm on any set of inputs for which they are not part of the entitled set. The second factor of $\left[1 + \left(\frac{X_i}{Y_i}\right)^{b_i}\right]^{a_i}$ can be calculated by dividing the values $m^{\gamma t} X_i^{b_i}$ and $m^{\gamma t} Y_i^{b_i}$ and then exponentiating with a_i. These values are the two values that are summed when a viewer calculates the session key when they are in the entitled set.

Since the value $m^{\gamma t}C$ is independent of any viewer-specific variable it means that for two values a_i and a_j the values $[m^{\gamma t}C]^{a_i}$ and $[m^{\gamma t}C]^{a_j}$ have the same base and if it happens that $\gcd(a_i, a_j) = 1$ the textbook RSA attack can easily be applied to derive the session key $m^{\gamma t}C$.

3.7.7 Security of the Scheme Without Access to a Broadcast

It has been shown that our scheme is insecure once a pair of attackers has been in an entitled set at least once. This does not show that security does not hold when attackers have never been in the entitled group. Recall that Pointcheval [32] proves that finding $(a+1)^e \bmod N$ from $a^e \bmod N$ and finding a from $a^e \bmod N$ and $(a+1)^e \bmod N$ together are equivalent to solving the RSA problem. These results seem similar to what the attacker in our scheme tries to achieve when attempting to calculate $\left(m^{\gamma t}X_i^{b_i} + m^{\gamma t}Y_i^{b_i}\right)^{a_i}$ when knowing only $\left(m^{\gamma t}X_i^{b_i}\right)^{a_i}$ and $\left(m^{\gamma t}Y_i^{b_i}\right)^{a_i}$.

3.8 A Possible Winner: Using Blinding Exponents

3.8.1 Design choices

The previous design tried to defend against the textbook RSA attack by having each viewer get keys that had different bases for each viewer but with exponents still known to all viewers. The problem with the design was that at the end the calculated keys had to be brought to the same base in order that each entitled viewer calculates the same key. This is the weakness that is exploited by the textbook RSA attack.

The next design tries to defend against the textbook RSA attack by obfuscating the exponents from the viewer in a similar way as the white-box implementations described in Section 2.4. Previously the viewers would have access to all the a_i values but in this design the exact values of these would be hidden through the use of blinding exponents. Without knowing the a_i exponents an attacker cannot mount the textbook RSA attack. They can apply the attack to the blinding exponents but this should not produce any useful information to the attacker. We first describe the system.

3.8.2 Overview

The entire system can be represented as a directed graph, as shown in Figure 3.2. A unique instance of such a graph is given to each viewer at setup time. Each viewer j is assigned a node in the graph labelled j. There are three additional nodes in the graph: the start, intermediate and end nodes. These nodes are labelled S, I and E, respectively. Note also that viewer i has no node associated with itself in the graph they receive.

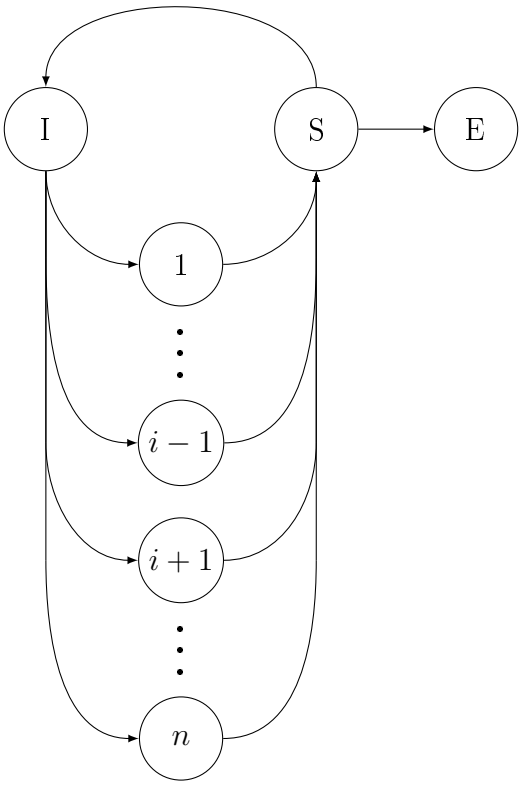


Figure 3.2: Graph of viewer i with a total of n viewers

A viewer derives a shared session key for a group $\mathbb{S}$ by traversing through all the nodes, except their own, associated with the viewers in $\mathbb{S}$. Because of this movement in the graph we define a position vector $\underline{P}_\ell$ for each viewer where the subscript ℓ indicates the number of transitions they have made so far.

As a viewer transitions between two nodes x and y in the graph they calculate a new position vector using the equation $\underline{P}_{\ell+1} = \mathcal{C}(x, y, \underline{P}_\ell)$ for some transition function $\mathcal{C}$ that is defined later. It is only possible to transition in one direction, as the edges clearly indicate.

3.8.3 Setup(λ, n):

3.8.3.1 The System Parameters $\mathcal{S}$

Generate an integer $N = pq$, where p and q are primes with $\|p\| = \|q\| = \frac{\lambda}{2}$. Again, λ is the security parameter. Each node x is then assigned values a_x and b_x with $a_x \leftarrow \mathbb{Z}^*_{\phi(N)}$ and $a_x \cdot b_x = 1 \bmod \phi(N)$, essentially an RSA key-pair. These key pairs must be distinct from one another, thus there exists no $x \neq y$ such that $a_x = a_y$. We also set $\gamma = a_E$, the a_i value of the end node. A base value m is chosen with $m \leftarrow \mathbb{Z}^*_N$.

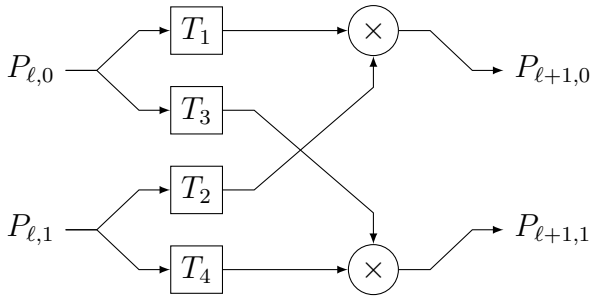


Figure 3.3: The transition function $\mathcal{C}$

3.8.3.2 The Viewer Parameters $\mathcal{P}_i$

Each viewer is given the modulus N as well as their own instance of the graph. This means that the transition function $\mathcal{C}$ associated with an edge is not the same for two different viewers. From here on all newly defined variables are assumed to be unique to the viewer the parameters are being generated for. We describe the process of generating the parameters for viewer i.

First each node x, except the end node, is assigned a blinding vector $\underline{R}_x = \begin{bmatrix} r_{x,0} & r_{x,1} \end{bmatrix}$ with $r_{x,0}, r_{x,1} \leftarrow \mathbb{Z}^*_{\phi(N)}$. For the end node we set $\underline{R}_E = \begin{bmatrix} 1 & 0 \end{bmatrix}$. The viewer is given the vector $\underline{M}_i = \begin{bmatrix} (m^{r_{S,0}})^{a_i} \\ (m^{r_{S,1}})^{a_i} \end{bmatrix} \bmod N$.

Define the transition function $\mathcal{C}$ between two nodes x and y as a function that satisfies

$$\mathcal{C}\left(x, y, \underline{P}_\ell\right) = \underline{P}_{\ell+1} \pmod N \tag{3.46}$$

with $\underline{P}_\ell = \begin{bmatrix} m^{Qr_{x,0}} \\ m^{Qr_{x,1}} \end{bmatrix}$ and $\underline{P}_{\ell+1} = \begin{bmatrix} m^{Qa_y r_{y,0}} \\ m^{Qa_y r_{y,1}} \end{bmatrix}$ and some unknown Q. The crucial thing to note is that a transition to a node adds the node's a_i factor to the exponent. Since the transition function only requires the presence of the blinding factors in $\underline{P}_\ell$, the added a_i exponents will not be removed no matter how many times the viewer transitions.

Such a transition function $\mathcal{C}$ can be found by giving viewer i the transition matrix $\mathbf{T}^{(i)}_{x,y} = \begin{bmatrix} T_1 & T_2 \\ T_3 & T_4 \end{bmatrix}$ which is used to calculate:

$$\left(m^{Qr_{x,0}}\right)^{T_1} \left(m^{Qr_{x,1}}\right)^{T_2} = m^{Qa_y r_{y,0}} \pmod N \tag{3.47}$$

and

$$\left(m^{Qr_{x,0}}\right)^{T_3} \left(m^{Qr_{x,1}}\right)^{T_4} = m^{Qa_y r_{y,1}} \pmod N. \tag{3.48}$$

Figure 3.3 depicts the transition function where $\underline{P}_\ell = \begin{bmatrix} P_{\ell,0} \\ P_{\ell,1} \end{bmatrix}$ and boxes represent exponentiation.

This leads to the congruences

$$T_1 r_{x,0} + T_2 r_{x,1} = a_y r_{y,0} \pmod{\phi(N)} \tag{3.49}$$

and

$$T_3 r_{x,0} + T_4 r_{x,1} = a_y r_{y,1} \pmod{\phi(N)}. \tag{3.50}$$

For each matrix $\mathbf{T}_{x,y}^{(i)}$ in a graph choose $T_1, T_3 \leftarrow \mathbb{Z}_{\phi(N)}^*$ and then calculate T_2 and T_4 from (3.49) and (3.50) . These transition matrices are only calculated for the edges shown in Figure 3.2. For the transition from the start to the end node the bottom row of the matrix $\mathbf{T}_{S,E}^{(i)}$ is set to 0 since $\underline{R}_E = \begin{bmatrix} 1 & 0 \end{bmatrix}$, but the top elements are calculated normally.

3.8.4 Encrypt$(\mathbb{S}, \mathcal{S})$:

To entitle a set $\mathbb{S}$ the centre chooses a value $t \leftarrow \mathbb{Z}_{\phi(N)}^*$, broadcasts the value

$$D = t \cdot B_{\mathbb{S}} \cdot (b_S b_I)^{|\mathbb{S}|-1} \tag{3.51}$$

where $B_{\mathbb{S}} = \prod_{i \in \mathbb{S}} b_i$ and encrypts the content under the key $K = m^{\gamma t}$. The $(b_S b_I)^{|\mathbb{S}|-1}$ factor is included because to entitle $|\mathbb{S}|$ viewers a viewer will transition through the intermediate and start nodes $|\mathbb{S}| - 1$ times, adding their corresponding a_i factors to the exponent each time.

3.8.5 Decrypt$(\mathbb{S}, i, \mathcal{P}_i, D)$:

A user has a position vector $\underline{P}_\ell$ with initial value $\underline{M}_i$, thus $\underline{P}_0 = \underline{M}_i$. As they traverse through the nodes this position vector is updated by applying the transition function of an edge to it. A viewer will transition through $|\mathbb{S}| - 1$ nodes associated with viewers. This means they will also transition through the start and intermediate nodes the same amount of times. Thus the viewer will use the position vector $\underline{P}_{3|\mathbb{S}|-3}$ when transitioning from the start to the end node.

When transitioning from the start to the end node using $\underline{P}_{3|\mathbb{S}|-3}$ the viewer will calculate the vector

$$\underline{P}_{3|\mathbb{S}|-2} = \begin{bmatrix} m^{\gamma \cdot A_{\mathbb{S}} \cdot (a_S a_I)^{|\mathbb{S}|-1}} & 1 \end{bmatrix}^T \tag{3.52}$$

where $A_{\mathbb{S}} = \prod_{i \in \mathbb{S}} a_i$. This top element is then exponentiated with the broadcast factor to give

$$\left(m^{\gamma \cdot A_{\mathbb{S}} \cdot (a_S a_I)^{|\mathbb{S}|-1}}\right)^{t \cdot B_{\mathbb{S}} \cdot (b_S b_I)^{|\mathbb{S}|-1}} = m^{(a_S b_S)^{|\mathbb{S}|-1} (a_I b_I)^{|\mathbb{S}|-1} (A_{\mathbb{S}} B_{\mathbb{S}}) \gamma t} = m^{\gamma t} \pmod{N}. \tag{3.53}$$

3.8.6 A Probable Proof of Security

We now present the original proof of security. There are several flaws in the arguments of this proof and they will be pointed out.

3.8.6.1 Security of the Broadcast Factor

Any dishonest viewer in the network has access to the broadcasted value $t \cdot B_{\mathbb{S}} \cdot (b_S b_I)^{|\mathbb{S}|-1} \pmod{\phi(N)}$. This group element on its own leaks no information about $\mathbb{B}$ or k, where $\mathbb{B}$ is the set of all b_i in the system. This is true because multiplication by a group element chosen uniformly at random is equivalent to choosing the entire element uniformly at random. To prove this we will use the following lemma from [20]:

Lemma 38. *Let $\mathbb{G}$ be a finite group, and let $m \in \mathbb{G}$ be an arbitrary element. Then choosing random $g \leftarrow \mathbb{G}$ and setting $g' = m \cdot g$ gives the same distribution for g' as choosing random $g' \leftarrow \mathbb{G}$. I.e., for any $\hat{g} \in \mathbb{G}$*

$$\Pr[m \cdot g = \hat{g}] = \frac{1}{|\mathbb{G}|}$$

where the probability is taken over random choice of g.

Proof. Let $\hat{g} \in \mathbb{G}$ be arbitrary. Then $\Pr[m \cdot g = \hat{g}] = \Pr[g = m^{-1} \cdot \hat{g}]$. Since g is chosen uniformly at random, the probability that g is equal to the fixed element $m^{-1} \cdot \hat{g}$ is exactly $1/|\mathbb{G}|$. □

We can apply this lemma to our broadcast factor because $B_{\mathbb{S}} \in \mathbb{Z}^*_{\phi(N)}$ and $t \in \mathbb{Z}^*_{\phi(N)}$. By setting $B_{\mathbb{S}} \cdot (b_S b_I)^{|\mathbb{S}|-1} = m$ and $t = g$ it is clear that the lemma holds.

Problems with Proof

Lemma 38 proves that no information about the the b_i values are leaked to an attacker. However, it does not prove that the attacker gains no information about $\phi(N)$. The attacker not knowing $\phi(N)$ is the basis of security in RSA-based schemes. While it is not immediately apparent that the attacker can indeed learn $\phi(N)$ it is a crucial detail that this lemma does not address.

3.8.6.2 Proof of Security for Construction

We will now prove the security of our broadcast encryption scheme relative to the RSA encryption scheme. First we define the experiment $\mathrm{BE}_{\mathcal{A},\mathsf{Setup}}(\lambda, n, k)$:

Definition 39. $\mathrm{BE}_{\mathcal{A},\mathsf{Setup}}(\lambda, n, k)$

1. Run $\mathsf{Setup}(\lambda, n)$ to obtain output $\mathcal{P}$ and $\mathcal{S}$.

2. Choose $\mathbb{S} \leftarrow \{1, \ldots, n\}, |\mathbb{S}| = n - k$.

3. Set $(D, K) = \mathsf{Encrypt}(\mathbb{S}, \mathcal{S})$.

4. $\mathcal{A}$ is given $\mathcal{P}_{\bar{\mathbb{S}}} = \{\mathcal{P}_i \,|i \notin \mathbb{S}\}$, D, $\mathbb{S}$ and outputs K'.

5. The output of the experiment is defined to be 1 if $K' = K$, and 0 otherwise.

We say that an adversary $\mathcal{A}$ breaks the collusion resistance of our scheme if

$$\mathrm{BE}_{\mathcal{A},\mathsf{Setup}}(\lambda, n, k) = 1 \tag{3.54}$$

for any $k < n$.

We will now show that if such an $\mathcal{A}$ exists such that $\mathrm{BE}_{\mathcal{A},\mathsf{Setup}}(\lambda, n, k) = 1$, we can construct an adversary $\mathcal{A}'$ that can efficiently break the RSA encryption scheme.

The idea is to choose a set of a_i values so that all the viewers in the collusion have an a_i that is a multiple of the RSA public key e, while the a_i of those in the entitled set are relatively prime to e. With these inputs $\mathcal{A}$ will produce a value m^X where X is relatively prime to e. $\mathcal{A}'$ can then use this value to recover the encrypted message m.

Definition 40. The adversary $\mathcal{A}'$:

On input (e, N) and $c = m^e$ do the following:

1. Choose a suitable n and $\mathbb{S} \leftarrow \{1, \ldots, n\}$.

2. Set $a_S, a_E \leftarrow \mathbb{Z}_N$ such that $\gcd(a_S, e) = \gcd(a_I, e) = 1$. A fairly large value for a_S ($\|a_S\| \backsimeq \lambda/2$) with many factors is preferable.

3. Choose $r_{\bar{\mathbb{S}}} = \{r_i \,| r_i \leftarrow \mathbb{Z}_N \; \forall i \in \{1, \ldots, n\} \setminus \mathbb{S}\}$.

4. Set $a_{\mathbb{S}} = \{a_i \,| a_i \leftarrow \mathbb{Z}_N, \gcd(a_i, e) = 1 \; \forall i \in \mathbb{S}\}$.

5. Set $a_{\bar{\mathbb{S}}} = \{a_i \,| a_i = e \cdot r_i \; \forall r_i \in r_{\bar{\mathbb{S}}}\}$.

6. Set $\langle m_i \rangle_{\bar{\mathbb{S}}} = \{r \,| r = c^{r_i} = m^{er_i} = m^{a_i} \; \forall r_i \in r_{\bar{\mathbb{S}}}\}$. These will later be turned into the vectors $\langle \underline{M}_i \rangle_{\bar{\mathbb{S}}}$ when the exponents $r_{S,0}$ and $r_{S,1}$ have been chosen for each i.

7. Construct $\left\langle \mathbf{T}^{(i)} \right\rangle_{\bar{\mathbb{S}}}$, $\langle \underline{M}_i \rangle_{\bar{\mathbb{S}}}$ and γ such that it represents a valid set of transfer matrices, including $\gcd(\gamma, e) = 1$. This construction is shown in Section 3.8.6.3.

8. Set $D = 1$ and $\mathcal{P}_{\bar{\mathbb{S}}} = \left(\left\langle \mathbf{T}^{(i)} \right\rangle_{\bar{\mathbb{S}}}, \langle \underline{M}_i \rangle_{\bar{\mathbb{S}}}\right)$.

9. Run $\mathcal{A}(\mathcal{P}_{\bar{\mathbb{S}}}, D, \mathbb{S})$ to obtain K'.

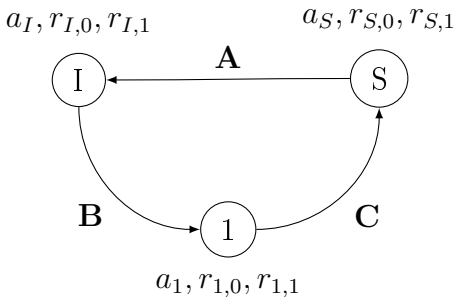


Figure 3.4: A cycle in the graph containing one viewer

What is the value of K'? First notice that if $D = 1$ then

$$\begin{aligned} D = &\quad (b_S b_I)^{|\mathbb{S}|-1} B_{\mathbb{S}} t = 1 \\ \Rightarrow \quad t = &\quad A_{\mathbb{S}} (a_S a_I)^{|\mathbb{S}|-1} \pmod{\phi(N)}. \end{aligned} \tag{3.55}$$

We know that on input $\mathcal{P}_{\bar{\mathbb{S}}}, D, \mathbb{S}$ the algorithm $\mathcal{A}$ outputs $K' = K$. In this case it would be

$$K' = K = m^{\gamma t} = m^{\gamma (a_S a_I)^{|\mathbb{S}|-1} A_{\mathbb{S}}}. \tag{3.56}$$

Because $\gcd(A_{\mathbb{S}}, e) = 1$, $\gcd(a_I, e) = 1$, $\gcd(a_S, e) = 1$ and $\gcd(\gamma, e) = 1$ it holds that $\gcd\left(\gamma (a_S a_I)^{|\mathbb{S}|-1} A_{\mathbb{S}}, e\right) = 1$. This means integers X and Y exist such that

$$X \left(\gamma (a_S a_I)^{|\mathbb{S}|-1} A_{\mathbb{S}}\right) + Ye = 1. \tag{3.57}$$

Thus $K'' = (K')^X \cdot c^Y = \left(m^{\gamma (a_S a_I)^{|\mathbb{S}|-1} A_{\mathbb{S}}}\right)^X \cdot (m^e)^Y = m^{X\left(\gamma (a_S a_I)^{|\mathbb{S}|-1} A_{\mathbb{S}}\right)\gamma + Ye} = m$ and the encrypted RSA message has been successfully recovered.

We have shown that our scheme is strong relative to the RSA encryption scheme. If an adversary exists that can break our scheme then the RSA scheme is insecure as well and since RSA has not been broken, we claim our system is secure as well. We will now describe the step where the necessary $\left\langle \mathbf{T}^{(i)} \right\rangle_{\bar{\mathbb{S}}}$ matrices are constructed.

3.8.6.3 Constructing the T Matrix

The only step left in the proof is to show that a set of matrices $\left\langle \mathbf{T}^{(i)} \right\rangle_{\bar{\mathbb{S}}}$, containing only positive integers, can be constructed that represents a valid set of transition values for the chosen a_i values.

Figure 3.4 shows a cycle in the graph with only one viewer. **A**, **B**, **C** are the transition matrices and the parameters associated with each node is shown next to it. Each matrix

$\mathbf{X}$ is defined as $\mathbf{X} = \begin{bmatrix} X_1 & X_2 \\ X_3 & X_4 \end{bmatrix}$. For the cycle to be valid the following equations must hold:

$$r_{S,0}A_1 + r_{S,1}A_2 = a_I r_{I,0} \tag{3.58}$$
$$r_{S,0}A_3 + r_{S,1}A_4 = a_I r_{I,1} \tag{3.59}$$
$$r_{I,0}B_1 + r_{I,1}B_2 = a_1 r_{1,0} \tag{3.60}$$
$$r_{I,0}B_3 + r_{I,1}B_4 = a_1 r_{1,1} \tag{3.61}$$
$$r_{1,0}C_1 + r_{1,1}C_2 = a_S r_{S,0} \tag{3.62}$$
$$r_{1,0}C_3 + r_{1,1}C_4 = a_S r_{S,1} \tag{3.63}$$

At this point only a_S, a_I and a_1 have been fixed. Adding (3.58) and (3.59) yields

$$r_{S,0}\left(A_1 + A_3\right) + r_{S,1}\left(A_2 + A_4\right) = a_I\left(r_{I,0} + r_{I,1}\right). \tag{3.64}$$

Choose $y, z, r_{S,0}, r_{S,1} \in \mathbb{Z}^+$ such that $\left(r_{S,0}y + r_{S,1}z\right) | a_S$. Because $\gcd(a_S, e) = 1$ it implies that $\gcd\left(r_{S,0}y + r_{S,1}z, e\right) = 1$ and we can set $\mathbf{T}_{S,E}^{(i)} = \begin{bmatrix} x & y \\ 0 & 0 \end{bmatrix}$ and $\gamma = r_{S,0}y + r_{S,1}z$.

Setting $A_1 + A_3 = a_I y$ and $A_2 + A_4 = a_I z$ gives

$$\begin{aligned} a_I\left(r_{I,0} + r_{I,1}\right) &= r_{S,0}\left(a_I y\right) + r_{S,1}\left(a_I z\right) \\ &= a_I\left(r_{S,0}y + r_{S,1}z\right) \end{aligned} \tag{3.65}$$

which implies that $\left(r_{I,0} + r_{I,1}\right) = \left(r_{S,0}y + r_{S,1}z\right) \Rightarrow \left(r_{I,0} + r_{I,1}\right) | a_S$, a property that will be useful later on.

The next set of steps must be repeatedly applied to each node in the graph representing a viewer.

Consider the following two equations:

$$r_{I,0}B_1 + r_{I,1}B_2 - a_1 r_{1,0} = 0, \tag{3.66}$$
$$r_{I,0}B_3 + r_{I,1}B_4 - a_1 r_{1,1} = 0. \tag{3.67}$$

A solution given by [14] is $B_1 = B_2 = B_3 = B_4 = a_1$ and $r_{1,0} = r_{1,1} = r_{I,0} + r_{I,1}$. Substituting the latter into (3.62) and (3.63) to solve for $\mathbf{C}$ yields:

$$\left(r_{I,0} + r_{I,1}\right)\left(C_1 + C_2\right) = a_S r_{S,0}, \tag{3.68}$$
$$\left(r_{I,0} + r_{I,1}\right)\left(C_3 + C_4\right) = a_S r_{S,1}. \tag{3.69}$$

Because $(r_{I,0} + r_{I,1}) | a_S$ and are all positive integers, the positive integer entries for $\mathbf{C}$ can easily be found.

After these steps have been followed enough times a valid set of transfer matrices have been constructed, finishing the proof.

Problems with Proof

In the proof, when constructing the T matrices the contents of the matrices is taken from $\mathbb{Z}^+$ and not $\mathbb{Z}^*_{\phi(N)}$. This changes the distribution of the values from what a real instance of the system would look like and thus the proof is faulty.

3.8.7 Cryptanalysis

The security flaw in this design comes from the congruences (3.49) and (3.50). They are

$$T_1 r_{x,0} + T_2 r_{x,1} = a_y r_{y,0} \pmod{\phi(N)} \tag{3.70}$$

and

$$T_3 r_{x,0} + T_4 r_{x,1} = a_y r_{y,1} \pmod{\phi(N)}. \tag{3.71}$$

The only variables known to an attacker in this context are T_1, T_2, T_3, T_4 taken from the $\mathbf{T}^{(i)}_{x,y}$ matrix. Observe that these equations can be rewritten as

$$\mathbf{T}^{(i)}_{x,y} \begin{bmatrix} r_{x,0} \\ r_{x,1} \end{bmatrix} = a_y \begin{bmatrix} r_{y,0} \\ r_{y,1} \end{bmatrix} \pmod{\phi(N)} \tag{3.72}$$

which leads to a transition cycle from the start node being written as

$$\mathbf{T}^{(i)}_{S,I} \mathbf{T}^{(i)}_{I,j} \mathbf{T}^{(i)}_{j,S} \begin{bmatrix} r_{S,0} \\ r_{S,1} \end{bmatrix} = \mathbf{T}^{(i)}_{j} \begin{bmatrix} r_{S,0} \\ r_{S,1} \end{bmatrix} = a_I a_j a_s \begin{bmatrix} r_{S,0} \\ r_{S,1} \end{bmatrix} = \alpha_j \begin{bmatrix} r_{S,0} \\ r_{S,1} \end{bmatrix} \pmod{\phi(N)}. \tag{3.73}$$

Letting $\mathbf{T}^{(i)}_j = \begin{bmatrix} A_j & B_j \\ C_j & D_j \end{bmatrix}$ creates the following two congruences:

$$A_j r_{S,0} + B_j r_{S,1} = \alpha_j r_{S,0} \pmod{\phi(N)} \tag{3.74}$$

and

$$C_j r_{S,0} + D_j r_{S,1} = \alpha_j r_{S,1} \pmod{\phi(N)} \tag{3.75}$$

with A_j, B_j, C_j and D_j known to an attacker. Multiplying the first congruence with $r_{S,1}$,

the second with $r_{S,0}$ and subtracting yields

$$\begin{aligned} & A_j r_{S,0} r_{S,1} + B_j r_{S,1}^2 - C_j r_{S,0}^2 - D_j r_{S,0} r_{S,1} && (3.76)\\ = \;& (A_j - D_j)\, r_{S,0} r_{S,1} + B_j r_{S,1}^2 - C_j r_{S,0}^2 && (3.77)\\ = \;& 0 \pmod{\phi(N)}. \end{aligned}$$

Similarly the attacker can find

$$(A_k - D_k)\, r_{S,0} r_{S,1} + B_k r_{S,1}^2 - C_k r_{S,0}^2 \;=\; 0 \pmod{\phi(N)}. \tag{3.78}$$

Multiplying $(A_k - D_k)$ with (3.77) and $(A_j - D_j)$ with (3.78) and subtracting will yield an equation of the form

$$(A_j - D_j)\left(B_k r_{S,1}^2 - C_k r_{S,0}^2\right) - (A_k - D_k)\left(B_j r_{S,1}^2 - C_j r_{S,0}^2\right) = 0 \pmod{\phi(N)} \tag{3.79}$$

which can be rewritten as $\beta_1 r_{S,1}^2 + \gamma_1 r_{S,0}^2$ with β_1, γ_1 known. Using two more $\mathbf{T}_j^{(i)}$ matrices the attacker can find $\beta_2 r_{S,1}^2 + \gamma_2 r_{S,0}^2 = 0 \bmod \phi(N)$. Multiplying and subtracting again will yield

$$(\beta_2\gamma_1 - \beta_1\gamma_2)\, r_{S,0}^2 = 0 \pmod{\phi(N)}. \tag{3.80}$$

Recall that $r_{S,0} \in \mathbb{Z}^*_{\phi(N)}$, which implies that $(\beta_2\gamma_1 - \beta_1\gamma_2) = 0 \pmod{\phi(N)}$ with $\beta_1, \beta_2, \gamma_1, \gamma_2$ all known to the attacker. Thus the attacker knows a multiple of $\phi(N)$ unless $(\beta_2\gamma_1 - \beta_1\gamma_2) = 0$ over the integers. Knowing this multiple of $\phi(N)$ allows an attacker to efficiently factor the modulus N [6]. Once the modulus has been factored an attacker can calculate the value of $\phi(N)$. Knowing the value of $\phi(N)$ does not imply that an attacker can find the values of the blinding factors, but it is possible.

This value of $\phi(N)$ is used to solve the following system of congruences:

$$\begin{aligned} A_j r_{S,0} + B_j r_{S,1} &= \alpha_j r_{S,0} \pmod{\phi(N)} && (3.81)\\ C_j r_{S,0} + D_j r_{S,1} &= \alpha_j r_{S,1} \pmod{\phi(N)} && (3.82)\\ A_k r_{S,0} + B_k r_{S,1} &= \alpha_k r_{S,0} \pmod{\phi(N)} && (3.83)\\ C_k r_{S,0} + D_k r_{S,1} &= \alpha_k r_{S,1} \pmod{\phi(N)}. && (3.84) \end{aligned}$$

Start by noting

$$\begin{aligned} B_j r_{S,1}^2 &= C_j r_{S,0}^2 + D_j r_{S,0} r_{S,1} - A_j r_{S,0} r_{S,1} \pmod{\phi(N)} && (3.85)\\ B_k r_{S,1}^2 &= C_k r_{S,0}^2 + D_k r_{S,0} r_{S,1} - A_k r_{S,0} r_{S,1} \pmod{\phi(N)} && (3.86) \end{aligned}$$

which leads to

$$\begin{aligned}
& B_kC_jr_{S,0}^2 + B_kD_jr_{S,0}r_{S,1} - B_kA_jr_{S,0}r_{S,1} = B_jC_kr_{S,0}^2 + B_jD_kr_{S,0}r_{S,1} - B_jA_kr_{S,0}r_{S,1} \\
\iff & B_kC_jr_{S,0} + B_kD_jr_{S,1} - B_kA_jr_{S,1} = B_jC_kr_{S,0} + B_jD_kr_{S,1} - B_jA_kr_{S,1} \\
\iff & (B_kC_j - B_jC_k)\, r_{S,0} = (B_jD_k + B_kA_j - B_jA_k - B_kD_j)\, r_{S,1} \\
\iff & r_{S,0} = \tfrac{(B_jD_k+B_kA_j-B_jA_k-B_kD_j)}{(B_kC_j-B_jC_k)} r_{S,1}. \qquad (3.87)
\end{aligned}$$

This still does not give a concrete solution to what $r_{s,0}$ and $r_{s,1}$ are, but when solving the system of equations note that if $\{r_{S,0}, r_{S,1}, \alpha_j, \alpha_k\}$ is a solution then so is $\{w \cdot r_{S,0}, w \cdot r_{S,1}, \alpha_j, \alpha_k\}$ for any $w \in \mathbb{Z}^*_{\phi(N)}$. This means that an attacker can choose any value in $\mathbb{Z}^*_{\phi(N)}$ for $r_{S,1}$ and then calculate $r_{S,0}$ from (3.87) before finding α_j and α_k via simple substitution.

3.9 Conclusion

In this chapter two collusion resistant broadcast encryption schemes taken from literature was presented. The algorithms needed to construct the schemes were given and shown to be correct. The assumptions from which these schemes derive their security were also given.

We then proceeded to design our own collusion resistant broadcast encryption scheme. Each iteration of the design was motivated and cryptanalysed and finally shown to be insecure. Each new iteration attempted to address the security flaw found in the previous design. Each of the schemes was shown to be susceptible to some variant of the textbook RSA attack except for the final scheme which leaks information about the modulus.

Having designed several schemes, the final proposed scheme will be implemented. Although the scheme is shown to be insecure against collusion attacks, we will compare our scheme against the two schemes taken from literature to show that our scheme does meet the bandwidth efficiency requirement. We chose to implement the final scheme as at that time it was not known that the scheme was insecure.

The next chapter describes the software design behind implementing such broadcast encryptions schemes as well as the framework needed to transform it into a product entitlement system and measure the performance of the schemes.

Chapter 4

Framework Implementation

4.1 Introduction

In this chapter we will discuss the design and implementation of a software framework within which broadcast encryption schemes can be tested and deployed as a working product entitlement system. Such a framework would allow for the easy deployment and testing of newly developed broadcast encryption schemes into a product entitlement environment. We first list the requirements of such a framework before discussing the conceptual design of the framework and finally detailing some practical problems encounters while implementing the framework.

4.2 Requirements

In the previous chapter we detailed some proposed designs for broadcast encryption schemes which can be used in a product entitlement system. These broadcast encryption schemes on their own do not constitute a working product entitlement system. To this end we build a product entitlement framework in which the broadcast encryption scheme used can be easily changed per broadcast. This framework will encrypt and transmit actual video content and which will be decrypted and displayed on client machines.

Because the system will have the ability to change the broadcast encryption scheme, the network architecture of the framework will be far removed from the actual implementation of the broadcast encryption scheme. This will allow for the development of new broadcast encryptions schemes without having to modify the existing framework to incorporate them.

Before deploying a new broadcast encryption scheme it must first be determined whether the performance of the scheme allows it to be used in a broadcast environment with changing entitlement sets and storage constraints. To make these tests easier to run a performance testing framework will be constructed that can report important

measurements, such as key calculation time, taken from the broadcast encryption scheme.

The following list formalises the requirements of such a framework:

- The broadcast encryption scheme must be interchangeable without having to rebuild the framework.
- The system must be able to stream video content which can be displayed on the receiver.
- The system must provide the ability to encrypt the data before sending it across the network.
- During the sending of a video stream the system must be able to add and remove receivers from the entitled group without interruptions to entitled viewers.
- The only customisation required to run performance tests on new broadcast encryption schemes is the specification of the test parameters.

Emulating the broadcast environment

In a true broadcast environment each set top box would receive the exact same broadcasted ciphertext stream. In a software simulated environment this can be achieved by IP multicasting or careful synchronisation of the network streams before they are encrypted. It has been decided that implementing either of these options would bring no real benefit to the system. Instead we will use several streams that encrypt using the same session key but have no synchronisation between them. As our implementation uses stream ciphers this will result in several ciphertext streams that were encrypted with the same key but differing plaintexts which is entirely insecure. As this configuration will only be used during testing this does not present a problem.

4.3 Network Software Architecture

One of the requirements of the system is that the client machines should be able to decrypt and display the video content that was broadcast by the server. Since rendering video is not part of the design of this thesis it was instead opted to use an existing video player. Our framework will interface with an existing video player through some network protocol supported by the player. This decision leads to the following framework: construct a proxy component that intercepts the outgoing network traffic of the video streaming server and encrypts it before sending it across the network. A similar proxy component receives the encrypted data on the client side and decrypts it before sending it to the rendering client to display the video. For the video data to be decrypted correctly these proxy components need to agree on a key and when to start using this key. To achieve

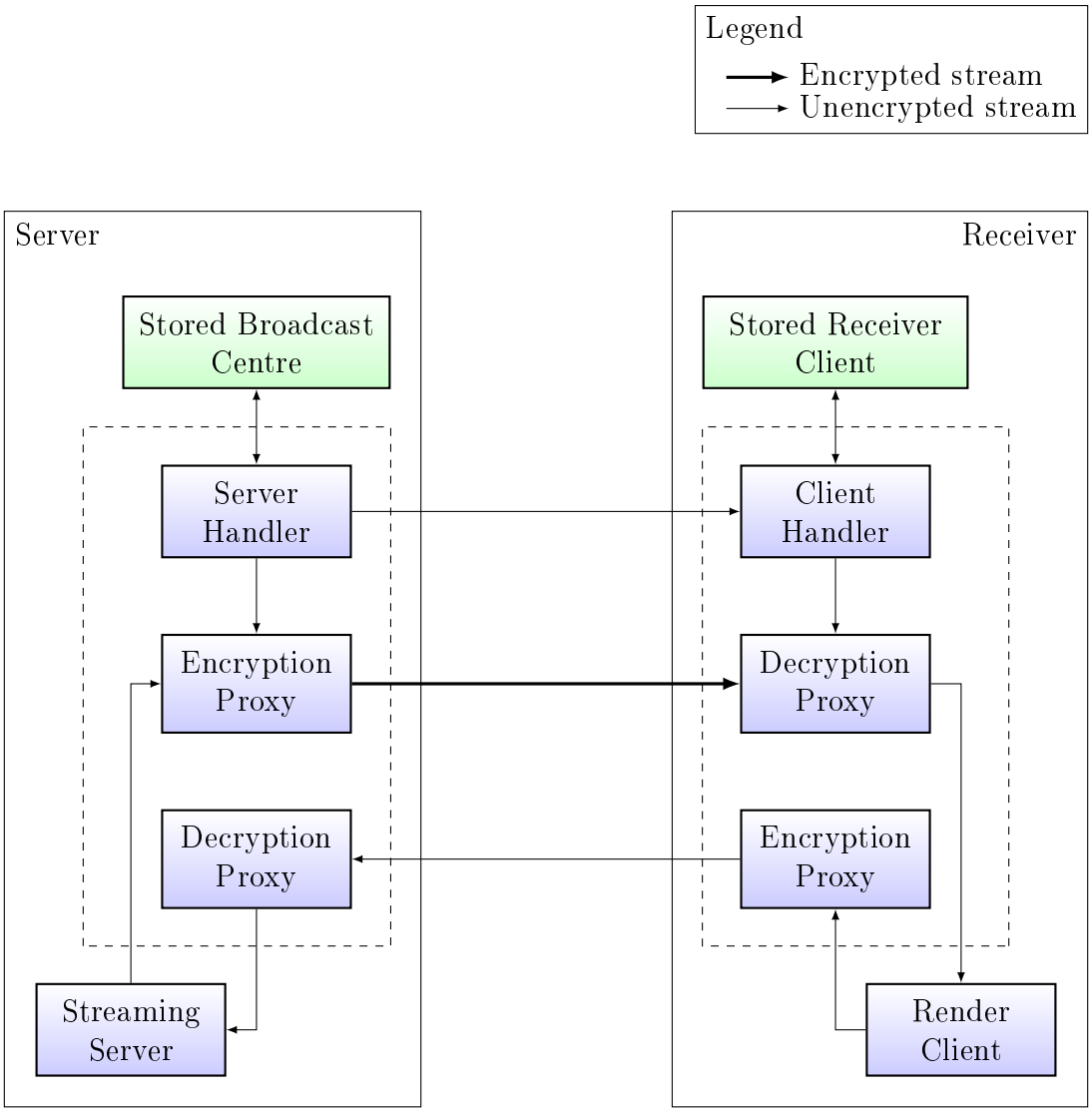


Figure 4.1: Software system architecture

this an additional handler component is constructed. On the client side the handler will be responsible for calculating the content encryption key from the key information sent to it by the server and passing that information to the proxy component. On the server side the handler component will calculate the encryption key according to the current entitled set and send the corresponding message header across the network unencrypted. This message header is sent unencrypted as this is part of the design requirements for a broadcast encryption scheme. When these components are tied together they are essentially simulating a conditional access client on a set-top box at the client side.

This framework is illustrated in Figure 4.1. The dashed boxes represent the collection of components constructed for this thesis. Note that one pair of proxy components do not have an encrypted stream between them and also transfers data from the receiver to the server, something that would not happen in a real broadcast environment. These components are included because some video rendering clients are not designed to run in

a true broadcast environment and can therefore send some identifying information to the video streaming server and the framework needs to pass this information along. Since this does not occur in a real broadcast environment we do not encrypt these data. Because the proxy component already supports redirecting of traffic we reuse this component without encryption to redirect the data from the render client to the streaming server.

We have given a general overview of the network architecture and will now describe in detail how each component works.

4.3.1 Server Handler

An important design specification was that the video rendering should be uninterrupted for any entitled viewer in the system. Calculating session keys might take up to several seconds depending on the broadcast encryption scheme used. This requires that the new session key should already be calculated by the time the encryption proxy has need of it. The responsibility of calculating the session key falls to the server handler.

For each receiver client that connects a new server handler is created. The server handler is mainly responsible for interacting with the stored broadcasting centre to generate the session keys and message headers, and communicating these keys to the correct components. The message header is broadcast in a different stream from the encrypted video data. This makes it easier to measure the bandwidth usage performance of the schemes than if the key information would have been interwoven with the encrypted video data in the same stream.

Once a request to modify the entitled set is given to the server handler it uses the stored broadcast centre to generate a new set of keys. Once these keys are calculated the server handler sends an entitlement message to the connected receiver client. It specifies the point where the new keys should be used to a user-specifiable number of bytes from the current position in the stream. This delay is introduced to give the receiver client enough time to receive the entitlement message as well as calculate the new session key. The most recent entitlement message is also stored so that if a new client connects during a broadcast that entitlement information can immediately be sent to them.

4.3.2 Client Handler

The client handler operates in much the same way as the server handler, but instead of broadcasting and creating the entitlement information it receives and consumes it. The client handler also runs in a different thread from the encryption proxy and interfaces with the stored receiver client to calculate the new session keys as the entitlement information is received. If the current receiver is not in the entitled set it will still request a session key from the stored receiver client. The key generated will not be the one needed to correctly

decrypt the secured content and will result in the encryption proxy sending random data to the video renderer.

4.3.3 Encryption Proxy

An encryption proxy operates on a set of streams. It reads data from one stream, encrypts those data and then writes the encrypted data to the second stream. The data can be encrypted by any symmetric cipher but for stream ciphers the encrypt and decrypt operations are the same and thus the same component can be used for both operations. Stream ciphers also have the ability to encrypt an arbitrary amount of bytes while block ciphers have to encrypt a multiple of the block size. Since the requests to change the session key are asynchronous to the encryption step, a request could arrive that requires the key to be changed in the middle of a block. This problem can be averted in two ways: use a padding scheme for the last block or synchronise the blocks with the session key change requests. Both of these approaches require more effort than using a stream cipher for encryption and therefore stream ciphers are preferred over block ciphers for our implementation.

4.3.4 Session Key Change Requests

If the entity running the broadcast centre decides to change the entitled set this change must be communicated to all the clients. The handlers on each side of the connection processes these requests. On the server side the handler requests a new session key and a new broadcast key. This is packaged together with the entitled set $\mathbb{S}$ and the mark at which the new key should be used into an entitlement message. This message is passed to the encryption proxy as it contains all the information the encryption proxy needs to correctly change the key.

4.3.5 Stored Broadcast Centre and Receiver Client

The broadcast centre and receiver clients are generated separately from the network framework. Instead they are generated beforehand and then each instance of the receiver clients are stored on the appropriate client machine. There are two main reasons for this:

1. generating the system parameters on a client machine is insecure and
2. the underlying broadcast scheme can be swapped out without touching the framework as per the design requirements.

These stored objects must conform to the design specifications of a broadcast centre and receiver client. More specifically, the broadcast centre must make available an $\mathsf{Encrypt}(\mathcal{S})$ method and the receiver client an $\mathsf{Decrypt}(\mathbb{S}, D)$ method. The signatures of these methods

differ from those defined for a broadcast encryption scheme in Section 3.1 as they lack any way of passing system or viewer parameters. These parameters are already instantiated in the objects when they are stored on disk and therefore there is no need to pass them to the methods again. This also means that each stored receiver client is irrevocably tied to a specific viewer identity.

4.4 Testing Framework

The testing framework consists of two separate parts. The first is used for pure performance testing. This measures the time and space needed for the broadcast encryption scheme to operate. The second part tests the networking code of the product entitlement system and that it works correctly with any broadcast encryption system it is given to use.

4.4.1 Performance Testing

The three main measurements that can be taken with a broadcast encryption scheme is the size on disk of the various stored components, the time the scheme requires to calculate new keys and the amount of network traffic needed to send message headers. This corresponds to the resources that Fiat and Naor [16] attempt to optimise. Examining the three methods that define a broadcast encryption scheme, from Section 3.1, we have

- $\mathsf{Setup}(\lambda, n)$
- $\mathsf{Encrypt}(\mathbb{S}, \mathcal{S})$
- $\mathsf{Decrypt}(\mathbb{S}, i, \mathcal{P}_i, D)$

and between them there are seven parameters that can be varied. But $\mathcal{S}$, $\mathcal{P}_i$ and i depend on the output of $\mathsf{Setup}(\lambda, n)$ and D depends on the output of $\mathsf{Encrypt}(\mathbb{S}, \mathcal{S})$. Thus the only parameters we are interested in varying are λ, n and $\mathbb{S}$.

We are interested in measuring the total amount of storage required by the broadcast centre and receiver clients. For the receiver clients the space needed will be averaged over all receiver clients in the population. We are also interested in the time required to calculate the session keys, both on the broadcast centre and on the individual receivers.

Since the testing for each broadcast encryption scheme will be the same up to setting the boundaries of the parameters to be tested, a module was created that takes as input the range of one of the parameters to test over, fixed values for other parameters and outputs the results in a table format. Some of the tests are reported on at the same time, for instance measuring time taken by the $\mathsf{Setup}(\lambda, n)$ method will also report on the size of the objects created by it.

4.4.2 Correctness Testing

To ensure that the network architecture works correctly the receiver clients must decrypt the content sent by the broadcasting centre when they are in the entitled set. To check for correct decryption each receiver client is given a reference file. This is also the file that will be broadcast and encrypted by the broadcast centre. As each client receives the entitlement information they calculate a new key based on that information, decrypt the incoming content with that key and check the decrypted content against the reference file they have been given. To report if each byte is correctly or incorrectly decrypted is unnecessary since we expect large segments of the broadcast to be continuously decrypted correctly or incorrectly seeing as the entitled set changes with a low frequency compared to the volume of data transferred. The assumption is made that there are no errors during the transmission of any data and thus that any decryption errors are due to the key being incorrect. Instead the component will only report when it goes from decrypting correctly to incorrectly and vice versa. A certain number of bytes must be sequentially decrypted incorrectly for the switch to be recognised. This is because a stream cipher has a $\frac{1}{256}$ probability of randomly correctly decrypting a byte even if an incorrect key is used. These data can then be easily checked against the entitlement messages sent for the broadcast to determine if the data is being decrypted correctly at the right time.

4.5 Practical Considerations

During the implementation of the frameworks described previously some implementation issues where encountered. In this section we will discuss each of these issues and how they were resolved.

4.5.1 Video Rendering

The choice was made to use VideoLan Client (VLC) as it provides the ability to act as both a video streaming server and rendering client. VLC is an open source software package and therefore the temptation exists to modify its network components to include the encryption code of this project. After some investigation it was decided to not to do this as it would require working with an unfamiliar codebase, an unfamiliar programming language as well as tying the project to the VLC code making it harder to integrate in other software packages. Instead we found after some experimentation that setting up the VLC server to stream via HTTP and intercepting and resending that stream still allows the VLC client to correctly render the video.

4.5.2 Server Handling

The server handler is run in a separate thread from the encryption proxy so that the key generation and message header broadcasting steps do not hinder data broadcasting. Similarly the client handler is run in a separate thread from the decryption proxy. It was mentioned that newly generated session keys only get used a user-specifiable number of bytes after the entitled set has changed. For our testing purposes a 54396kb file that contains 491 seconds of video is used. This translates into an average 111kb of data per second of video. Measurements in Section 5.2 show that message headers are mostly less than 1kb which will incur a negligible amount of transfer time when compared with the bandwidth required to transfer the video content. In our testing setup the new key is set to be used 500 000 bytes after the entitlement group request has been received. This gives the client $\frac{500000}{111 \cdot 1024} \backsimeq 4.4$ seconds to calculate the new key. A larger buffer might be needed if used in practice, but for correct synchronisation testing timely content delivery to the client is not a concern.

4.5.3 Encryption Proxy

As requests to change the session keys happen in an asynchronous manner and the encryption proxy runs in its own thread, it has to access the requests in a thread-safe manner. This is done by using a thread-safe queue from the standard language library. Any new requests for session key changes is added to the back of the queue while the encryption proxy only looks at the start of the queue. This operates under the assumption that the requests arrive in an ordered manner. If the queue used is a doubly linked list the amount of time each thread locks the other out from reading the object is minimal. Minimising the time that each thread locks the object is crucial as the encryption proxy blocks until it has peeked at the head of the queue. If it blocks for too long it will interrupt the video rendering of the receiver.

4.5.4 Session Key Changes

The object that contains the message header also contains a field for the session key to be used with this message header. When the server handler constructs the message header it stores the session key in this field. This object is also entered into the encryption proxy queue. The encryption proxy now knows the session key to use and at what point in the stream to start encrypting data with it. A clone of this object is then made by the server handler and the session key field is cleared before being sent across the network. Having the session key field present in the object allows the receiver client to fill in that field with whatever session key it calculates before being entered into the decryption proxy's queue.

4.5.5 Correctness Testing

In order to test the correctness of the decryption routines for large viewer populations several receiver client instances need to be run. To facilitate this we use Amazon EC2. Amazon EC2 allows for a large number of virtual machines to be instantiated as needed. Each of the virtual machines is loaded with an Amazon Machine Image (AMI). This image already contains the Linux operating system and all the required libraries and data needed to run a receiver client. When testing for large populations a broadcast centre is first created on a virtual machine. Once the broadcast centre has started, all the receiver client machines are created. In order for the receiver clients to know what the address of the broadcast centre is its IP address is passed as meta-data to all the receiver client machines. When all the receiver clients have connected to the broadcast centre the centre starts the broadcast and intermittently changes the entitled set. All receiver clients monitor these changes and after the broadcast is finished checks to see if it correctly decrypted all the content it was allowed to access. The result of this check is stored in an Amazon DynamoDB and only machines that have reported success are terminated. Receiver clients that experienced errors are kept online and investigated for errors.

4.6 Conclusion

In this chapter the software design of a product entitlement system was discussed as well as a framework for testing broadcast encryption schemes. We began the discussion by listing in detail the requirements of the systems followed by a brief discussion on the assumptions made about the broadcasting environment in which the network framework will be tested.

The design of the framework was given and discussed. This framework has been implemented and the implementation issues encountered were discussed. A motivation for using Amazon EC2 for testing the correctness of the schemes over a network was also given.

The purpose of the product entitlement system is to show that the broadcast encryption schemes can practically work and the goal of the testing framework is to measure the performance of the broadcast encryption schemes. These performance measurements can be used to make an informed decision on which broadcast encryption scheme is suited to a specific product entitlement system. In the next chapter we report and discuss the results of the performance and network tests.

Chapter 5

Experimental Results

5.1 Introduction

In the previous chapter we have described a framework used to measure performance of broadcast encryption schemes as well as using these schemes in a product entitlement system. In this chapter we present and discuss the results from this framework when tested on the two reference implementations mentioned in Sections 3.3 and 3.4 as well as our own scheme presented in Section 3.8.

In addition to determining if the product entitlement system works correctly we will be measuring the bandwidth usage, computation time and storage requirements of each of the schemes.

A Note on Comparing Key Sizes

The two reference implementations both derive their security from elliptic curve group assumptions while our scheme derives its security from the integer factorisation problem. It is well known that for comparable security elliptic curve schemes require smaller key sizes than integer factorisation based schemes. This is exemplified in the partial table taken from the NIST recommendation for key management [1], shown in Table 5.1.

Some of the tests measure results relative to the key size and to portray the result accurately we must plot the results relative to the bits of security offered by the key size as per Table 5.1. We calculated a function to map the security parameter λ to the bits of

Bits of security (Λ)	Integer field cryptography (λ)	Elliptic curve cryptography (λ)
80	1024	160
112	2048	224
128	3072	256
192	7680	384

Table 5.1: Equivalent security bits according to NIST SP 800-57

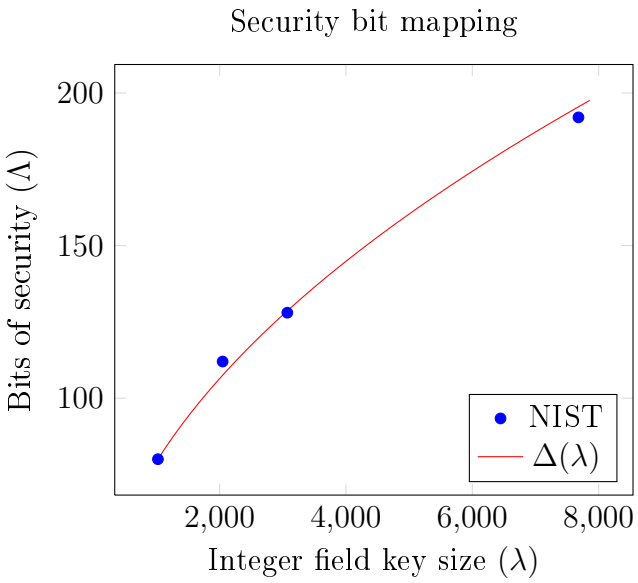


Figure 5.1: The key size mapping function

security Λ. This was done by fitting a polynomial function to the values found in Table 5.1. The function that will be used to map the integer field cryptography key sizes to the bits of security is $\Delta(\lambda) = 2.075\sqrt{\lambda} + 13.6$. This function is plotted against the data values from Table 5.1 in Figure 5.1 and it can be seen that the function closely approximates the NIST recommended key sizes. For elliptic curve cryptography the relation $\Lambda = \frac{\lambda}{2}$ is used.

5.2 Bandwidth efficiency

Bandwidth efficiency is an important design criterion for a product entitlement system as noted in Section 3.1. If a scheme uses too much bandwidth for entitlement purposes it becomes almost useless in a bandwidth constrained environment. While the absolute size of the entitlement messages is a concern it is also important to note the growth rate of the messages with respect to the input parameters, specifically the size of the entitlement set.

Motivation

As bandwidth is a scarce resource in a broadcast environment it is important to use it sparingly. Thus it is necessary to know how the size of the entitlement messages are affected by the various setup parameters of the schemes. Large messages will be expensive and slow to send. Messages that are slow to transmit to all viewers will have a detrimental effect on the viewers experience since they have to wait longer to gain access to the broadcast. The schemes' theoretical claims of $O(1)$ messages growth with respect to the size of $\mathbb{S}$ must also be verified.

Test configuration

The parameters that will be varied for these tests is the bits of security Λ, total viewer population n and size of the entitled set $|\mathbb{S}|$. For each test one parameter will first be varied while the other two parameters are fixed and subsequently the two parameters that influence the size of the messages will be varied. The test will set up each scheme in turn and generate 256 entitlement messages using the supplied parameters and report the median size of the entitlement messages as well as the standard deviation of the measurements. The viewers in the entitled set is chosen uniformly at random each time.

Results

Figure 5.2 and Figure 5.3 show the results of the measurements.

Discussion

The largest message header size measured was more than 1000 bytes. The maximum size of a typical entitlement management message is 256 bytes and thus will require the message header to be split across at least 4 entitlement management messages. This is not ideal and very inefficient. A network message created by our scheme with 80 bits of security and 1024 total viewer population contains one 1024 bit integer and one bit vector of length 1024. These two objects can be represented in 256 bytes, but the measurements at this point indicate the message is almost twice this size. Investigating in more detail the way that the network software stack used serialises the messages will shed some light on this large increase in message size.

Both of the elliptic curve based groups have a near constant size no matter what the size of the security parameter. As these schemes use finite fields with a size of 512 bits regardless of the security parameter the size of the element is not affected by the security parameter. As they both also contain two elements in the message header the size of the network messages for these schemes follow each other very closely across all measurements. Our scheme shows slight quadratic increase in size due to the integers used that grow quadratically with the required bits of security, as per the NIST standards. At 80 bits of security our scheme outperforms both the elliptic curve schemes but from Figure 5.3 it is clear that this is no longer the case when increasing the bits of security above 100.

All three schemes display a linear growth with regards to the total viewer population. This is to be expected as the number of group elements in the message stays the same, but the bit vector needed to represent the entitled set has to grow larger. There is a plateau around the 500 viewer population in the growth of the network message size. The bit vector used to represent the entitled set uses several 64-bit integers to represent the set. The two measurements in the plateau were made at 520 and 576 viewers, respectively. Representing 520 bits requires nine 64 bit integers, but these nine integers can also be

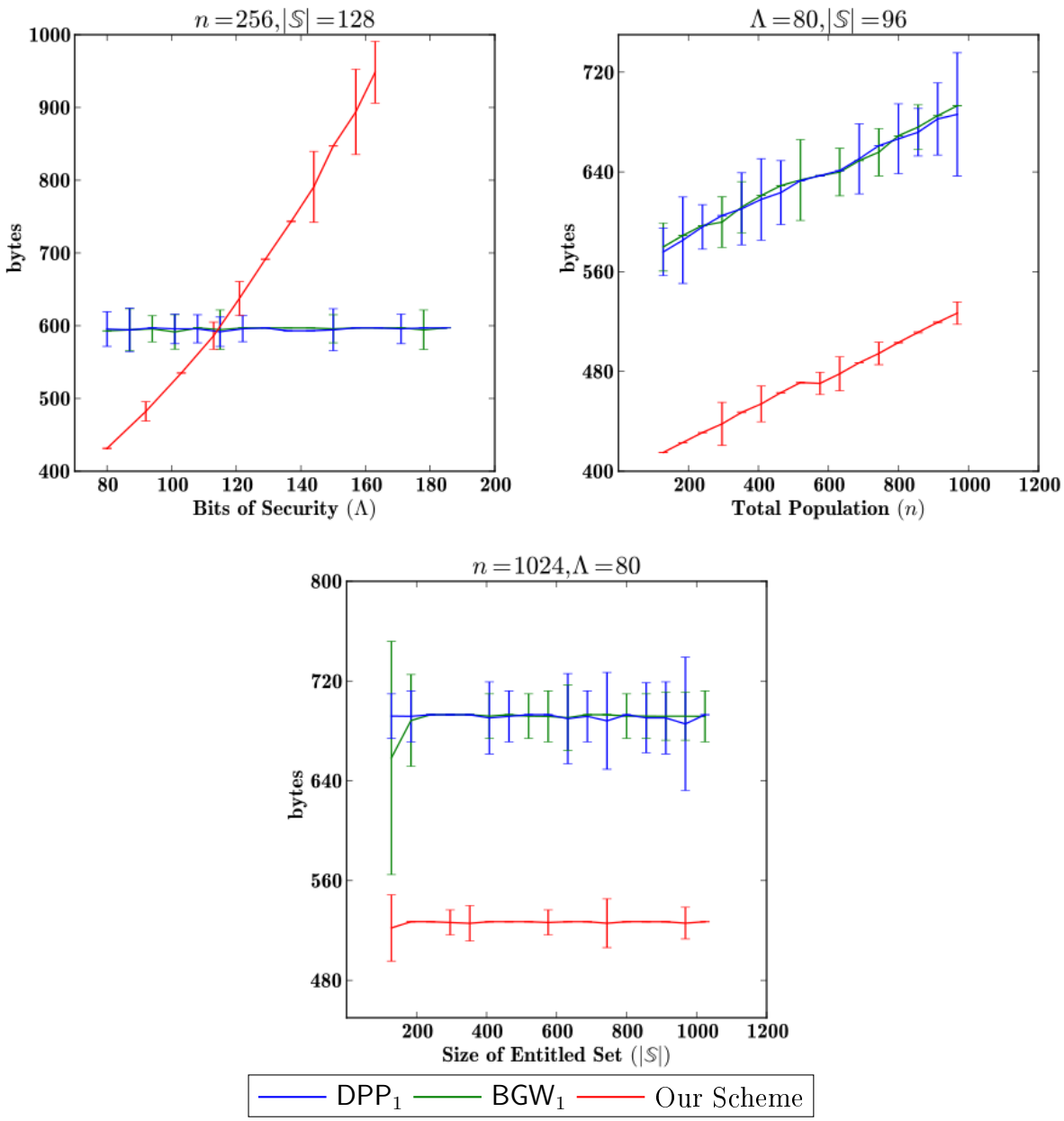
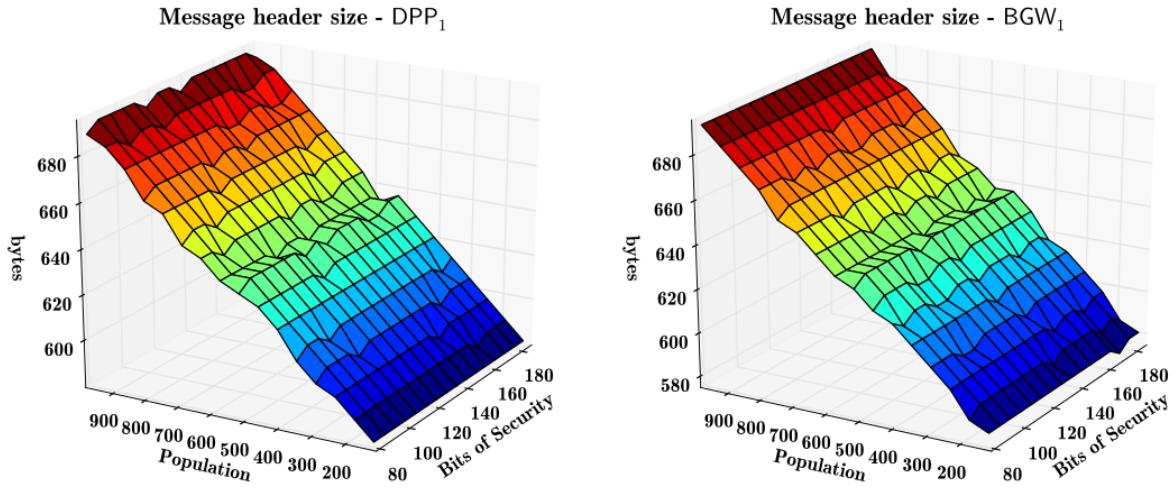


Figure 5.2: Standard deviation of network message size

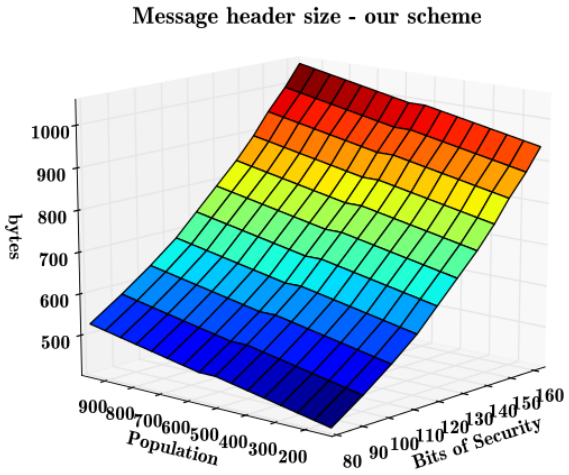
used to represent exactly 576 bits as well. Therefore the size of the representation does not grow between these two measurements as no new 64 bit integers have to be added to represent the new, larger population.

The size of the entitled set has almost no effect on the size of the network message as can clearly be seen from the zero growth exhibited by all three schemes when increasing the size of this set and thus all schemes are bandwidth efficient.



(a) DPP_1

(b) BGW_1



(c) Our scheme

Figure 5.3: Median of network message size across several parameters

5.3 Computation time

While bandwidth is the most constrained resource in a broadcast environment it is not the only limited resource. The time available to the broadcast centre and the set top box for cryptographic calculations is limited by the design requirement that the viewers enjoy an uninterrupted broadcast. Processors are also of limited power and can only do a finite amount of calculations in any given timespan. In this section we will investigate the time required by each of the broadcast encryption schemes tested to execute various functions.

5.3.1 Scheme setup times

Motivation

Before a product entitlement system can be deployed the necessary broadcast centre and receiver clients must first be generated. The test will only report the total time taken to generate a broadcast centre together with its receiver clients. The time it takes to generate each individual client is not reported as a system would only be deployed as a whole unit due to the static way the schemes are implemented.

Test configuration

The parameters that will be varied for these tests is the bits of security and total viewer population as the size of the entitled set does not influence the setup phase of the schemes. For each test one parameter will be varied while the other parameter stays fixed. Afterwards both parameters will be varied. The test will set up 16 instances of each scheme in turn and report on the median time and standard deviation taken to initialise the broadcast centre with all its receiver clients.

Results

Figure 5.4 and Figure 5.5 show the results of the measurements.

Discussion

Clearly both DPP_1 and BGW_1 outperform our scheme with regards to the setup time required. This is because with both DPP_1 and BGW_1 the clients receive only one unique group element while the rest of the information is shared across all viewers. In contrast to this all group elements given to a receiver in our scheme are unique to that receiver and need to be calculated.

Increasing the bits of security for our scheme results in a quadratic increase of the calculation time. This increase can be contributed to the size of the integers used that must increase quadratically in size relevant to the bits of security. The calculation time for our scheme also increases quadratically, albeit slightly slower, when increasing the total population. The number of calculations done per receiver in our scheme is linear in the total population as each node in a receiver's graph needs to be constructed and calculated. When increasing the population linearly this results in a quadratic overall increased computation time.

When compared to the setup times of DPP_1 and BGW_1 the near 17 minutes needed for the setup of our scheme with around 700 receivers seems daunting, but this is a once off costly operation. Once the scheme has been setup and deployed to set top boxes this operation does not need to be performed again. If the total viewer population increases

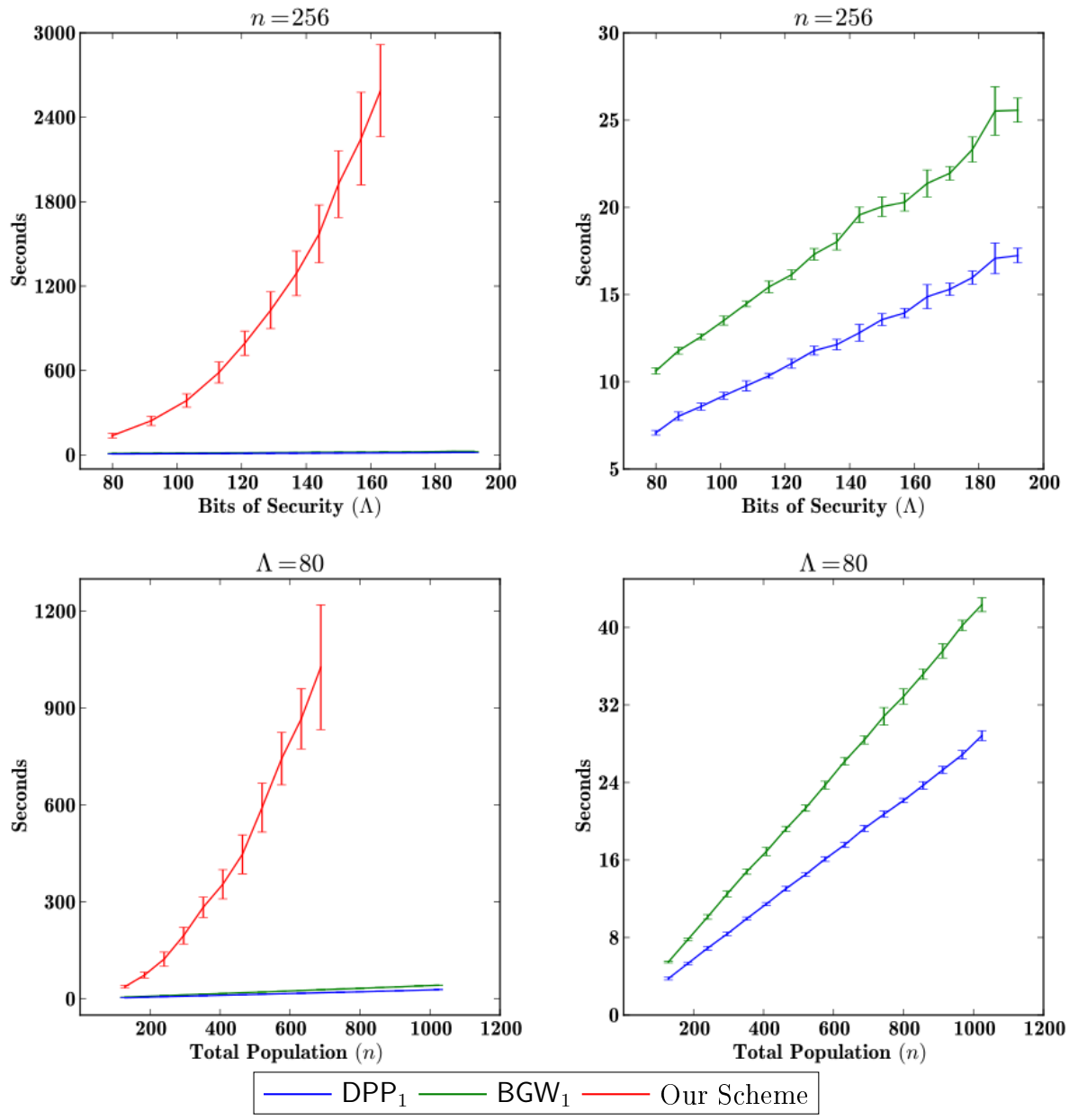
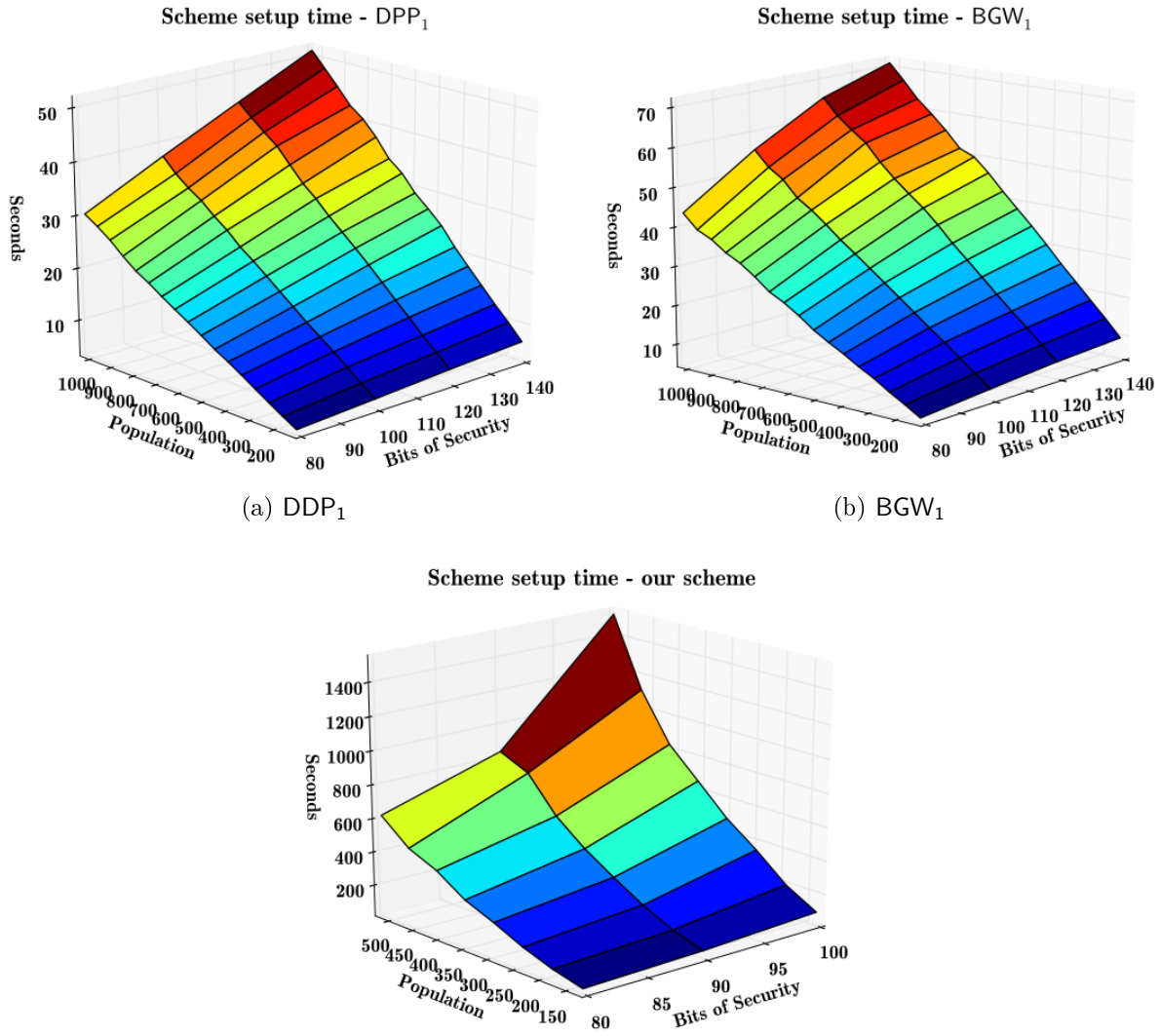


Figure 5.4: Standard deviation of scheme setup time

after a scheme has been deployed then the entire scheme will have to be set up again and redeployed. However, since the viewers are partitioned into groups of 1024, as discussed in Section 2.7, a new scheme with 1024 viewers can be set up to contain the new viewers. By using this method the old schemes do not have to be redeployed in the event that the total viewer population increases and we incur the time cost of setting up a new scheme only once for every 1024 viewers.



(a) DDP_1

(b) BGW_1

(c) Our scheme

Figure 5.5: Median of scheme setup time across several parameters

5.3.2 Generating Broadcast Keys

Motivation

The centre needs to generate a fresh key for each broadcast, but more importantly it needs to be able to generate new keys on the fly as to enable new viewers to join in the broadcast or cut off pirate viewers. If the generation of these keys are too time consuming it will lead to a negative customer experience. This test does not take into consideration the time it takes to broadcast the keys, only the time taken to generate them.

Test configuration

The test will vary one parameter while keeping all others fixed. The tests will generate 256 keys and report the median time to generate a key over these 256 attempts and

also report the standard deviation of the measurements. The viewers in the entitled set are chosen uniformly at random each time a key is generated. When varying the total population the test is slightly different for DPP_1. Instead of keeping the size of the entitled set constant the number of receivers not in the entitled set is kept constant. We know that theoretically the number of calculations that DPP_1 must do is related to the size of the revoked set. Thus if we want to show that the size of the population has no effect on the number these calculations the size of the revoked set must be kept constant.

Results

Figure 5.6 and Figure 5.7 show the results of the measurements.

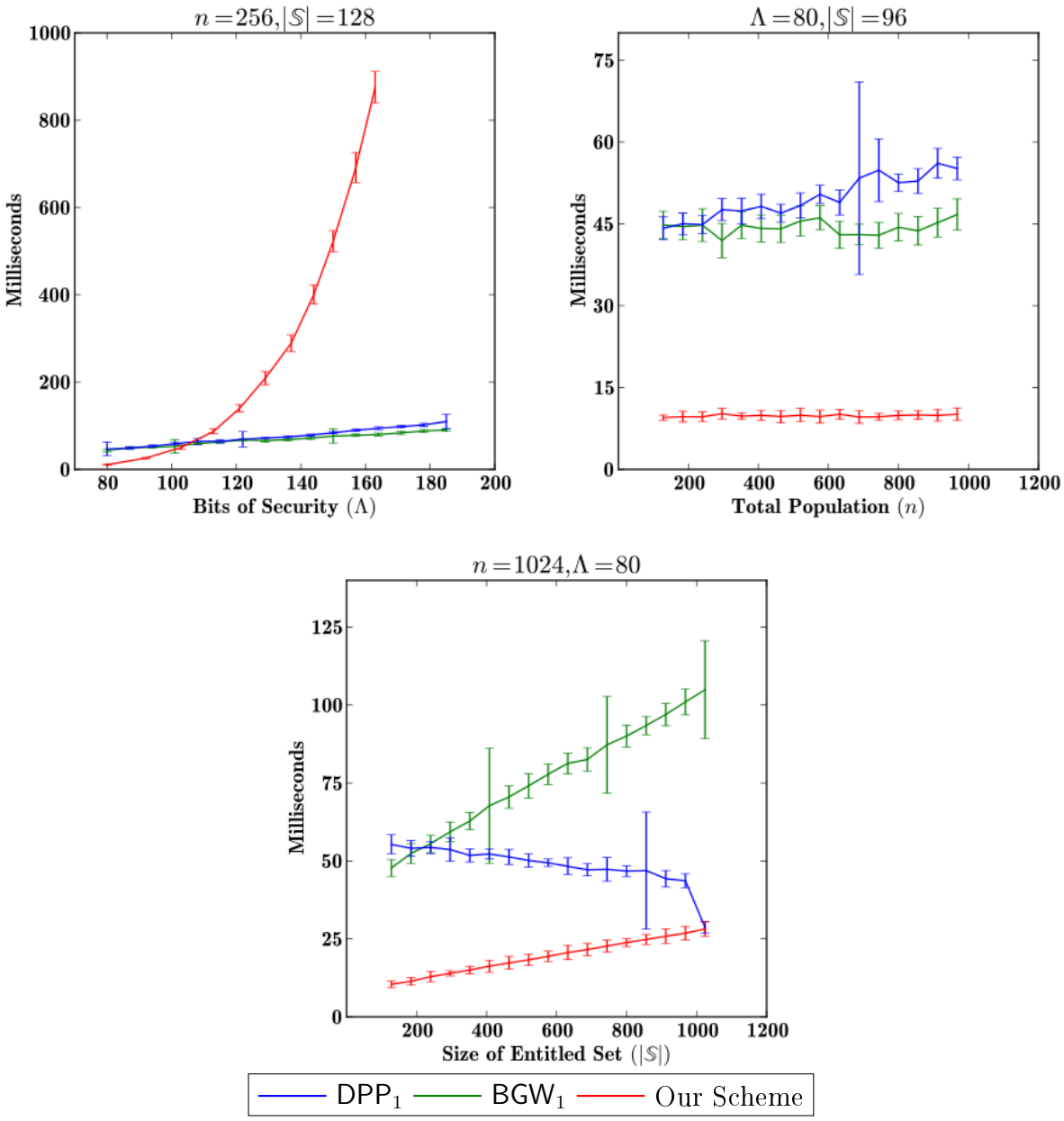
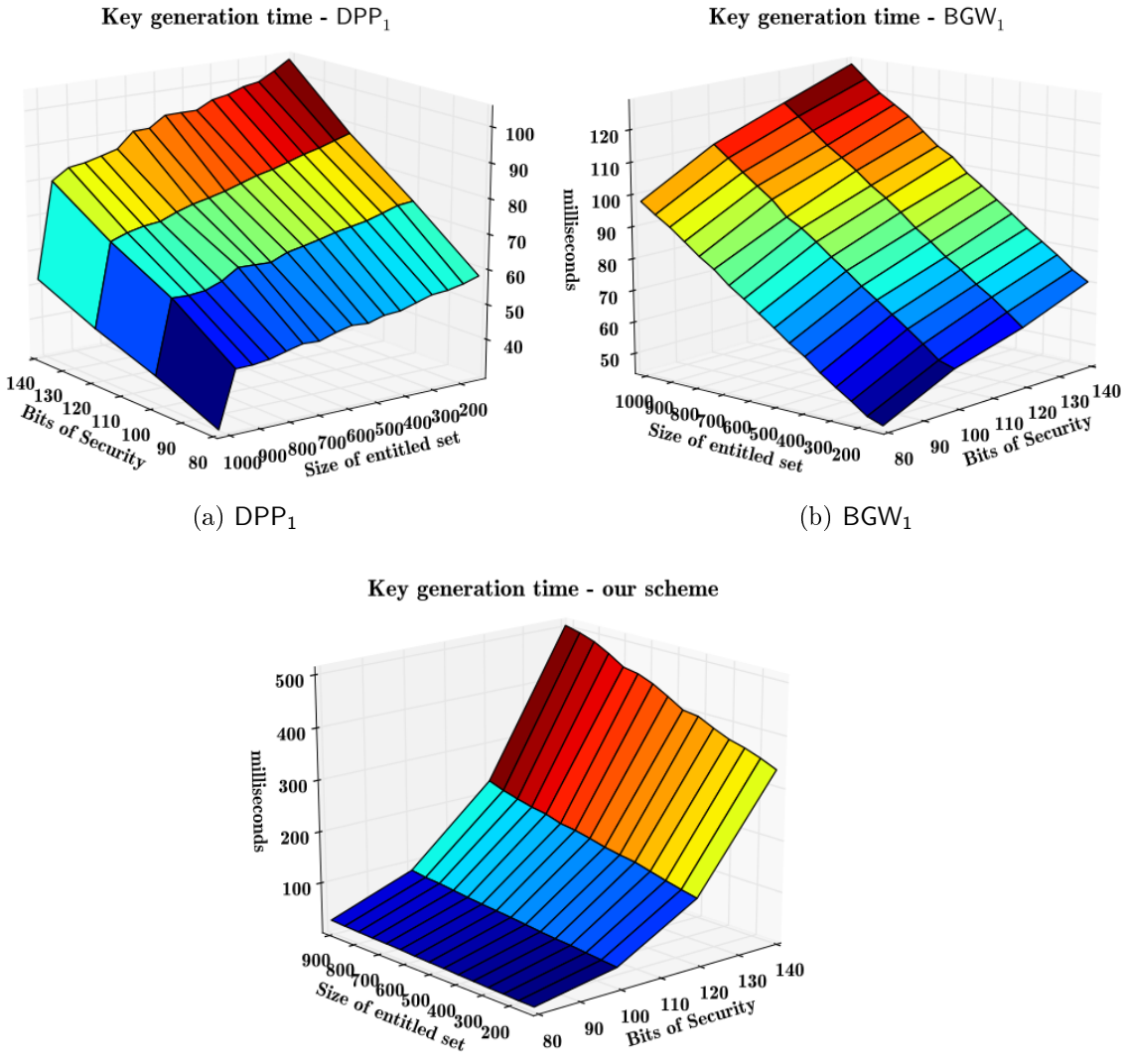


Figure 5.6: Standard deviation of message header generation time



(a) DPP_1

(b) BGW_1

(c) Our scheme

Figure 5.7: Median of message header generation time

Discussion

At 80 bits of security our scheme outperforms DPP_1 and BGW_1 due to the less computationally expensive multiplications it uses instead of having to add points on an elliptic curve. This advantage quickly disappears as the size of the integers used for our scheme increases quadratically in size relative to the number of bits of security. The time needed for BGW_1 and DPP_1 to generate keys stays almost constant and finally outperforms our scheme by a large factor.

The total population has no effect on the time needed to generate a broadcast key in any of the three schemes. This is expected as the size of the entitled set (or revoked set in the case of DPP_1) is the main contributing factor to key generation time at the centre.

Our scheme and BGW_1 both have a linear growth when increasing the size of the entitled set while DPP_1 shows a slight linear decline in generation time. Again at the 80

bits of security point our scheme requires less time per calculation than the elliptic curve schemes and thus shows a slower growth than BGW_1. The decline in the time needed by DPP_1 is due to the size of the revoked set shrinking which is the contributing factor in the generation time of the key. The sudden drop at the end occurs because if everyone is in the entitled set DPP_1 does a less costly exponentiation with 1 instead of an integer similar in size to the group order.

All five large standard deviation readings shown by DPP_1 and BGW_1 are due to a single reading of around 350 milliseconds during the testing phase. This reading was most likely caused by the Java virtual machine garbage collecting all the previously generated test messages.

5.3.3 Deriving session keys

Motivation

Upon receiving a new broadcasted key each receiver needs to calculate a new session key. This calculation has to be fast so that new viewers can access the content as early as possible. This information will also be useful to establish how long a buffer should be given before the centre switches over to the new session key.

Test configuration

The test will vary one parameter while keeping all others fixed. The tests will generate one broadcasted key and report the median time taken by 16 viewers in the entitled set to derive the session key. It will also report the standard deviation over the 16 measurements. Because some of the operations measured can take a long time to complete only 16 measurements are taken in order to complete the test in a timely manner. The viewers in the entitled set is chosen uniformly at random each time a key is generated. Again for DPP_1 the number of viewers not in the entitled set is kept constant when increasing the population.

Results

Figure 5.8 and Figure 5.9 show the results of the measurements.

Discussion

Our scheme outgrows both DPP_1 and BGW_1 once again as the bits of security increases. DPP_1 shows a definite linear increase relative to the security parameter while BGW_1 seems unaffected. The calculations used in DPP_1 to find the coefficients are done in integers modulo the group order. Since the group order becomes larger as the bits of security

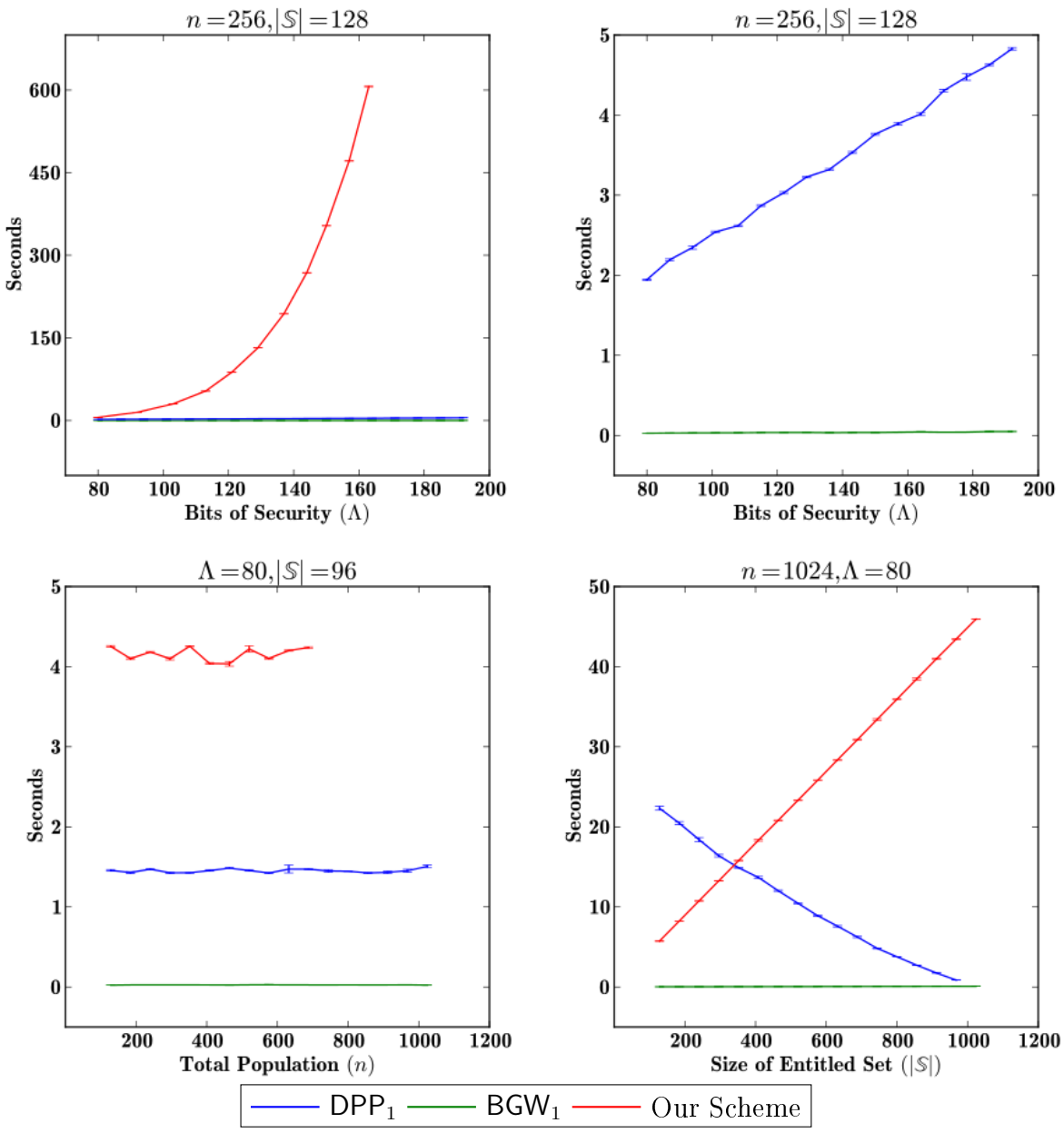
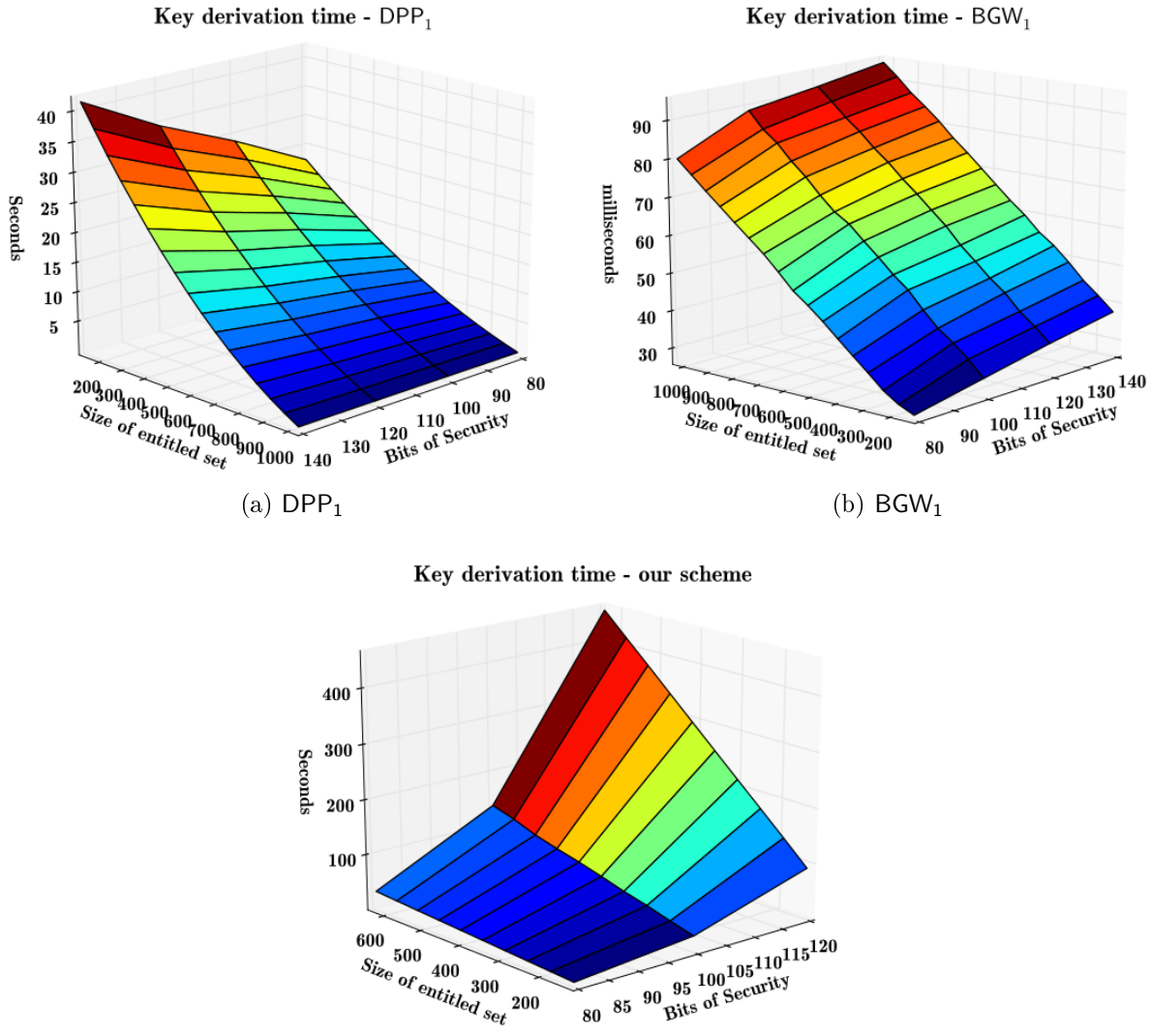


Figure 5.8: Standard deviation of key derivation time

increase, these calculations slow down as a larger group order results in larger integers being used in the calculations.

The time to derive a session key with all three schemes is unaffected by an increase in the total population. Here the $O(|\bar{\mathbb{S}}|^2)$ time algorithm used by DPP$_1$ can clearly be seen to be slower than the calculation when compared to the $O(|\mathbb{S}|)$ time algorithm used by BGW$_1$. Our scheme was measured at fewer points for the total population. While the operation to derive the session key itself is relatively quick, in order to make these measurements we must first set up a scheme with the required number of viewers. As this setup operation can take quite long as the total population increases, no measurements were taken at higher values for the total population.



(a) DPP_1

(b) BGW_1

(c) Our scheme

Figure 5.9: Median of key derivation time across several parameters

Due to the algorithm used we see a clear quadratic growth for DPP_1 in calculation time when deriving a session key as the the size of the entitled set decreases. Our scheme shows a linear growth while BGW_1 seems to stay constant as the set increases in size, but as can be seen form Figure 5.9b there is, in fact, a clear linear increase.

Comparison of techniques used for DPP_1

Delerablée et al. propose two algorithms [12] that are used to derive the session key when using DPP_1. These two algorithms are called Aggregate and Aggregate′. To derive a session key a receiver first runs Aggregate, which runs in $O(\left|\bar{\mathbb{S}}\right|^2)$ time, and then uses the output as input to Aggregate′, which runs in $O(\left|\bar{\mathbb{S}}\right|)$ time. Thus overall the time complexity of the originally proposed method to derive session keys is the same as our partial fractions based approach. A comparison of the time taken by both techniques to derive a session

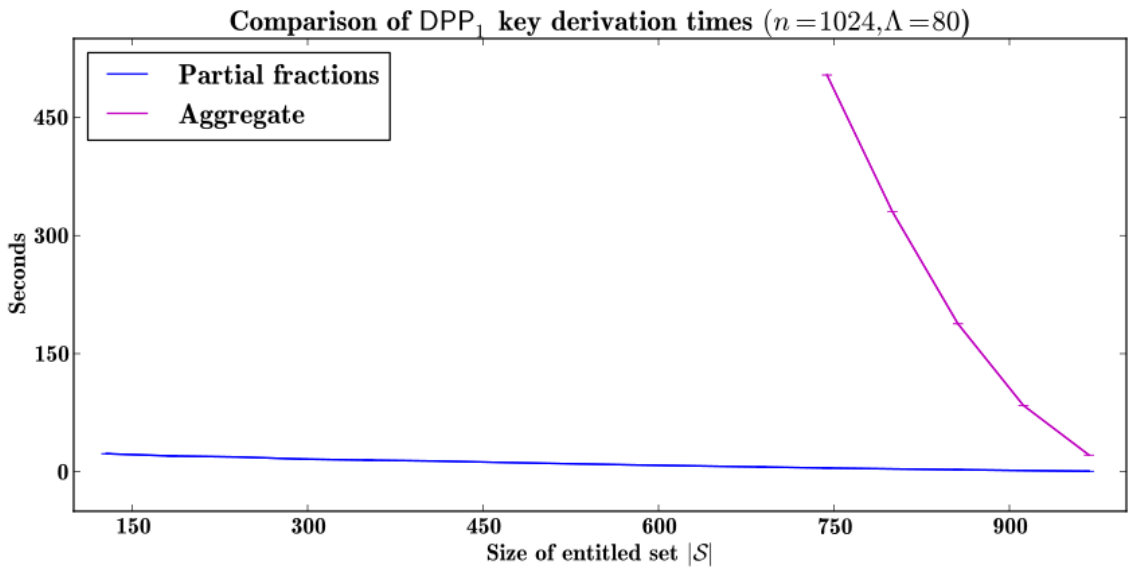


Figure 5.10: Comparison of time taken by different techniques used to derive a session key for DPP_1

key is shown in Figure 5.10.

From the graph our partial fractions based method can be seen to be significantly faster than the originally proposed methods. Due to the significant amount of time used by originally proposed method to derive a session key, it cannot satisfy both the requirements that a viewer gains quick access to a broadcast after being added to the entitled set and that viewers already in the set enjoy an uninterrupted viewer experience for small entitled sets. Our new technique allows for DPP_1 to satisfy both requirements and thus be used as a broadcast encryption scheme in situations where the size of the entitled set can be very small.

5.4 Storage space required on disk

Disk storage space is readily available to conditional access clients as modern day set top boxes can record several hours of video on disk. Due to the attack model in which the broadcast schemes are designed it does not compromise the security of the system if the conditional access client data are stored on unsecured hardware. But it is still a finite resource and care must be taken to ensure the clients and broadcast centres can store all the necessary data to operate correctly.

5.4.1 Broadcast centres

Motivation

Broadcast centres are likely to store information about all the receiver clients in order to be able to generate broadcast keys. This has the potential to become quite a large amount of data.

Test configuration

This test will in turn set up and then save to disk each of the broadcast centres for each scheme. The test will then report the size of these files on disk. The median size over 16 initialisations of a broadcast centre is reported and the standard deviation is also indicated.

Results

Figure 5.11 and Figure 5.12 show the results of the measurements.

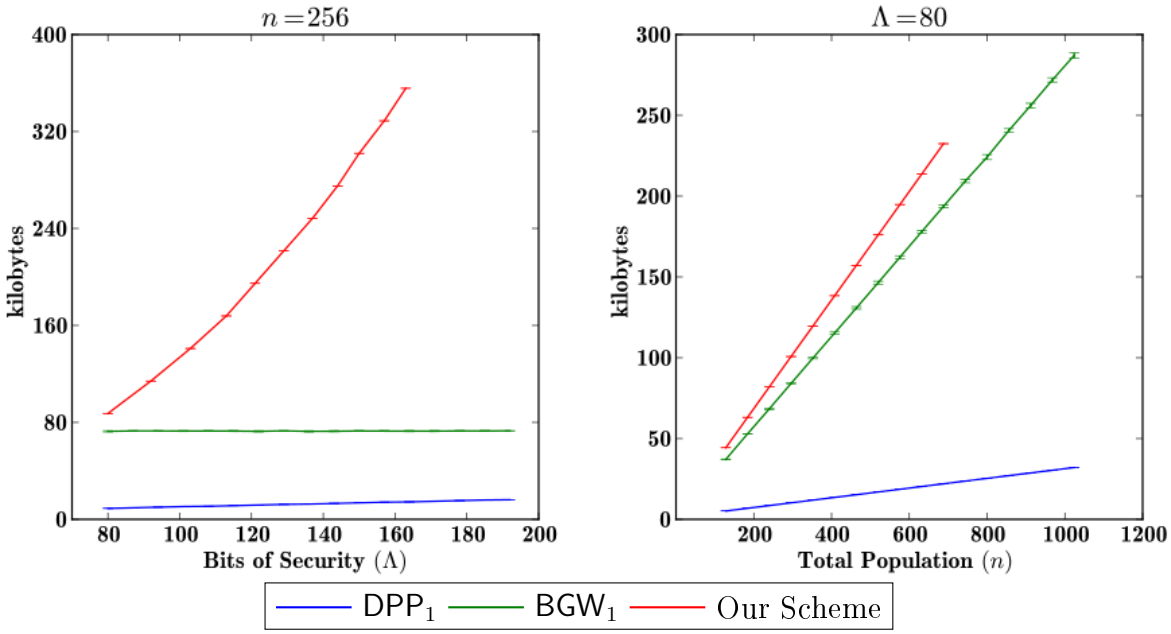
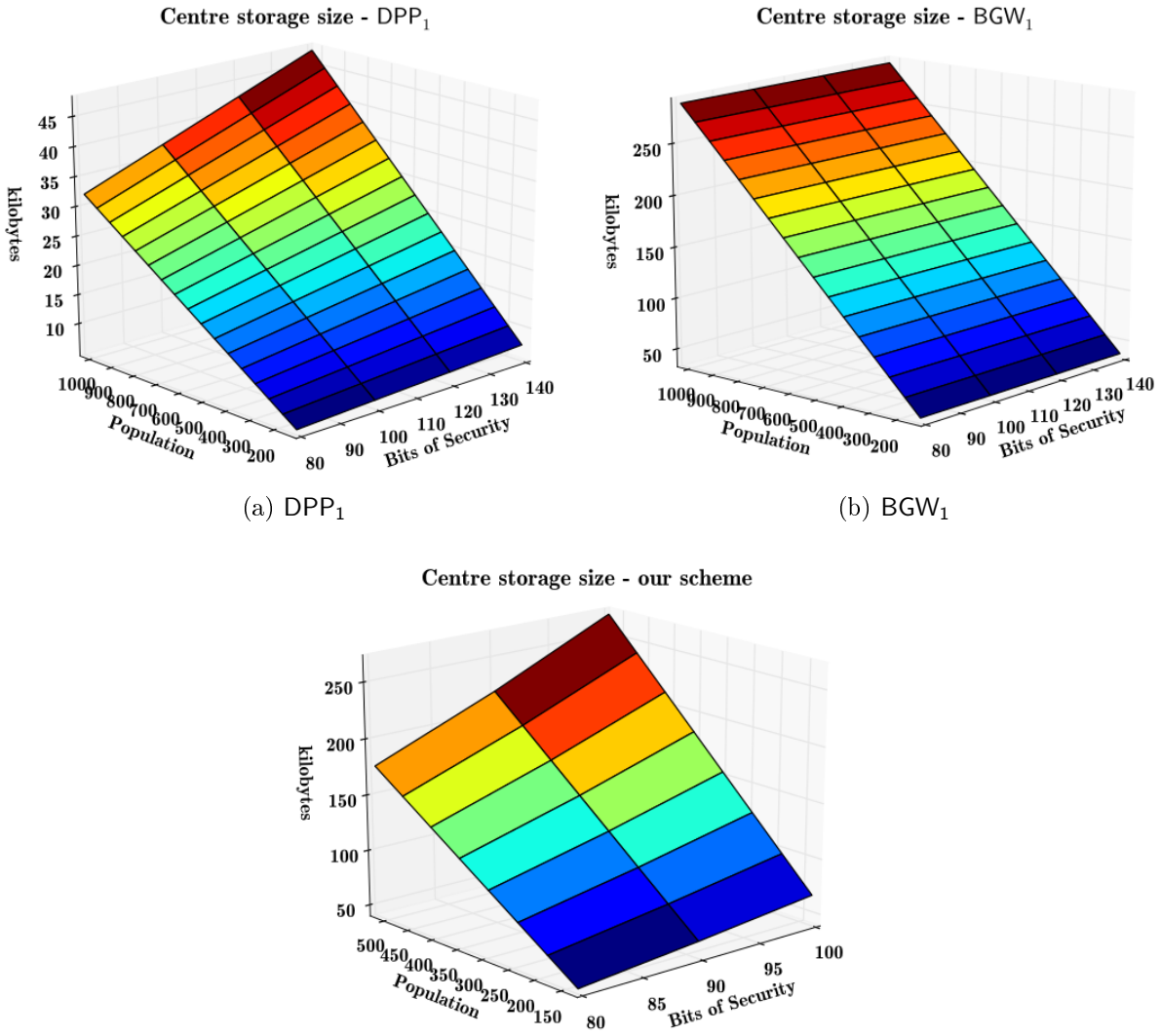


Figure 5.11: Standard deviation of broadcast centre storage size

Discussion

Only DPP_1 exhibits a linear growth in storage space when the security parameter is increased as it stores a single integer modulo the group order per receiver. As the group order grows larger with the increased bits of security, so do the integers associated with each receiver. BGW_1 shows no growth relative to the bits of security since it stores points



(a) DPP_1

(b) BGW_1

(c) Our scheme

Figure 5.12: Median of broadcast centre storage size across several parameters

on the elliptic curve which are always elements from a 512 bit finite field. Our scheme again displays a quadratic growth with respect to the security parameter which is caused by the quadratic growth in integer sizes used.

When keeping the bits of security fixed and increasing the total population all three schemes show linear growth. Here our scheme grows faster than the two elliptic curve based schemes. DPP_1 exhibits significantly less growth than BGW_1 as DPP_1 requires the centre to store only one integer associated with each receiver while BGW_1 requires storing two points on the elliptic curve.

5.4.2 Receiver clients

Motivation

The cost to manufacture a set top box increases with the storage space required to store the necessary data for the broadcast encryption scheme used. It is in the interest of any commercial operator to keep their cost to the client as low as possible. Also, if the broadcast centre ever decided to replace all the installed receiver clients it can be beneficial to have the amount of data that have to be transferred as low as possible.

Test configuration

This test will in turn set up all the receiver clients and then save 16 chosen uniformly at random to disk. The test then reports the median size of these saved receiver clients and the standard deviation of the measurements.

Results

Figure 5.13 and Figure 5.14 show the results of the measurements.

Discussion

BGW_1 shows no growth when increasing the security parameter. As it only stores points on the elliptic curve which has a 512 bit underlying field regardless of the size of the security parameter, this is to be expected. A quadratic growth is once again seen with our scheme as the size of the integers used for information storage increases quadratically. DPP_1 has a linear growth with regards to the bits of security as it stores integers modulo the group order, which grows as the bits of security increase.

All three schemes again exhibit a linear growth in storage space when increasing the total population, but now our scheme grows faster than the two elliptic curve based groups. Our scheme requires each receiver to store four group elements per other receiver in the system while the other two require each receiver to store two elements. DPP_1 stores one point on the elliptic curve and one integer modulo the group size and thus grows slower than BGW_1 which stores two points on the elliptic curve.

5.5 Product Entitlement system

In the previous sections we tested the performance metrics of the broadcast encryption schemes that have been implemented. The results from those tests provide a guideline on being able to choose the parameters for a scheme that will be deployed commercially. What the tests did not do was verify the correctness of the network architecture and that the product entitlement system does work. We will conduct those tests in this section.

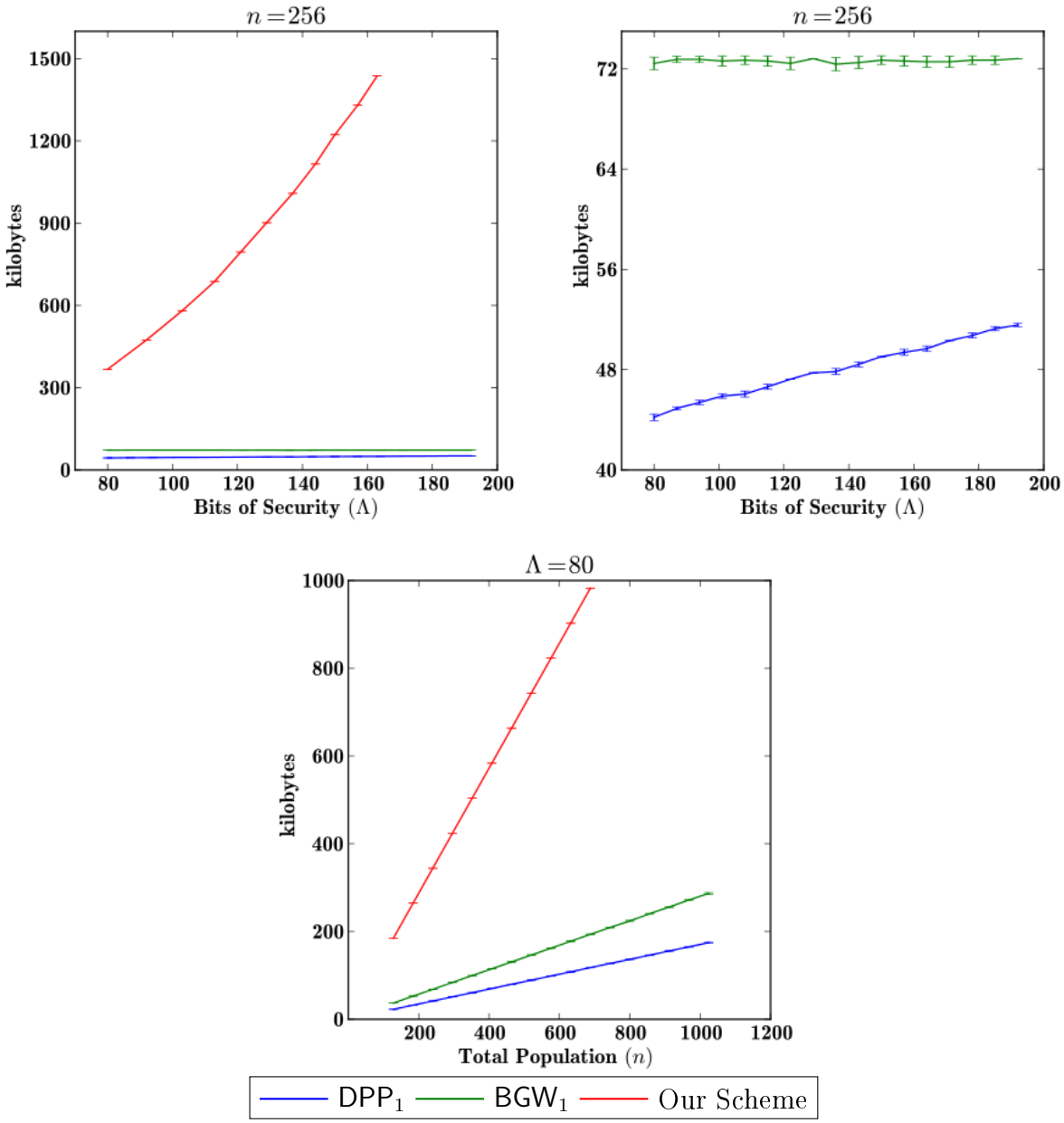
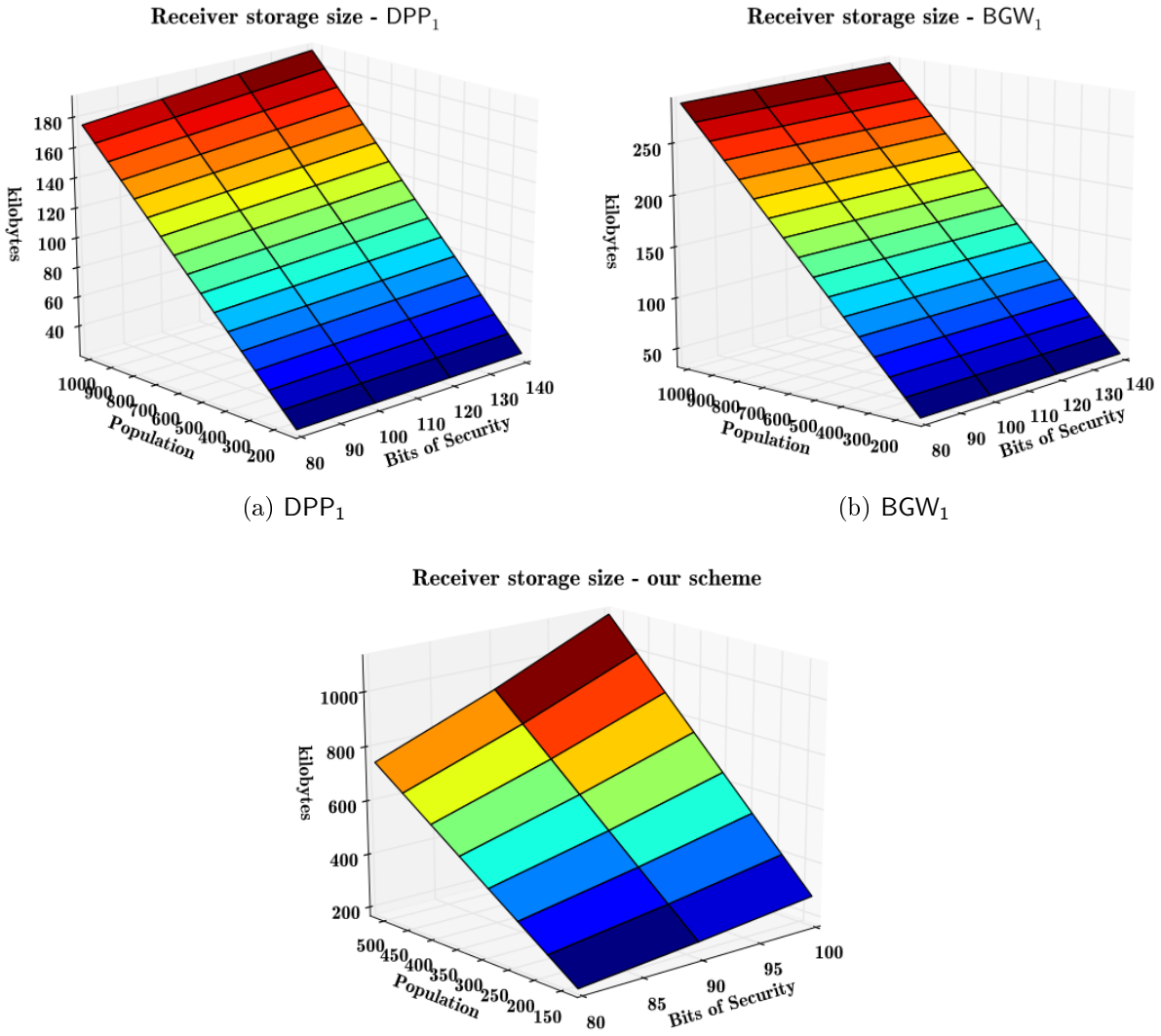


Figure 5.13: Standard deviation of receiver storage size

5.5.1 Correct Synchronisation of Decryption

Motivation

A product entitlement system would be useless if the system utilising it does not decrypt the protected video content correctly. While our simulation assumes that all data are transferred faultlessly, the encrypted and pseudorandom streams need to be synchronised correctly for decryption to be successful.



(a) DPP_1

(b) BGW_1

(c) Our scheme

Figure 5.14: Median of receiver storage size across several paramters

Test configuration

During the broadcast the broadcast centre will change the entitled set every 60 seconds. It will choose a random entitled set size and then random receivers to put in this set.

Each simulating receiver node is given the entire decrypted video before the start of the test. As the encrypted video is broadcast the receiver decrypts the video content and compares it to the unencrypted copy stored on disk, noting where the received encrypted data were correctly decrypted. After the broadcast completes, the client compares the received network messages to the noted correctly decrypted sections and checks if the content was decrypted correctly only at the time the receiver was supposed to be able to do so.

Each of the three implemented broadcast encryptions schemes will be tested. Each test will be run with schemes that contain 1024 users, but only a small subset of all

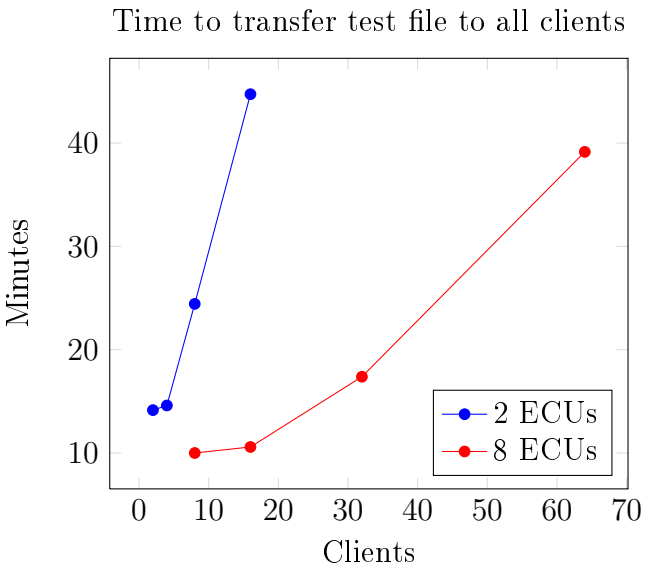


Figure 5.15: Time to complete synchronisation test

receivers will be simulated.

Results

All the tests that were run completed successfully. Figure 5.15 shows the total time taken by the server to completely transfer the test file to all connected clients.

Discussion

The framework correctly sends and decrypts the test file at the correct times, but the server infrastructure used does not scale well. As can be seen from Figure 5.15 there is a linear increase in the amount of time the server takes to transfer the entire file to all clients as the number of connected clients increases beyond a certain point. The period of negligible growth for low numbers of connected clients is due to the implementation detail that the network test cannot complete in less than 10 minutes. This is to give the server enough time to change the entitled set several times during the test. Increasing the number of Elastic Compute Units (ECUs) helps to slow the growth, but for both sets of measurements as soon as an ECU has to service more than two clients the time to complete the test increased significantly. To emulate a full 1024 receivers a system with 512 ECUs will have to be used, unfortunately Amazon does not offer such a system. A distributed system could be considered where intermediate servers duplicate and relay the information from the initial server, but currently the framework is unable to relay entitlement messages which is required to test the receivers.

The framework is therefore unable to deliver content in a timely manner to a large amount of test clients. However, in a true broadcasting environment the server would only generate one stream of encrypted data that is broadcast to all clients. Thus our

framework will work in a true broadcast environment.

5.5.2 Switching of Underlying Cryptographic Schemes

Motivation

Our system was designed and constructed with the idea in mind that new broadcast encryption schemes can be tested on it. For this reason it must work if the underlying scheme is switched out for a different one. This also applies to the stream cipher used for the encryption step.

Test configuration

The same approach will be taken as with the synchronisation test but at a smaller scale. The system was not designed to switch out the underlying scheme during run time and thus for each test the broadcast scheme and the stream cipher will be set before the test is run. Since the total number of configurations of the broadcast encryption schemes and stream ciphers is low enough we chose to test all the combinations. We will test all three implemented broadcast encryption schemes in conjunction with AES in OFB mode and RC4.

Results

All tests completed successfully.

Discussion

The test showed that our framework can successfully broadcast and decrypt content with various combinations of cryptographic schemes. This allows the framework to switch to newer schemes as old schemes become more insecure.

5.6 Conclusion

In this chapter we evaluated the performance of the implemented broadcast encryption schemes across several parameters. The results were then analysed and used to compare the schemes. The results also validate the theoretical predictions made of the bandwidth requirements of the schemes. The significant reduction in computation time for DPP_1 when using a technique we developed was also shown.

We also verified that the network architecture synchronises the decryption of the content correctly and adheres to the design requirement that the underlying cryptographic schemes that are used can be changed for each broadcast. We also discussed the inability of the network testing server to scale to a large number of clients.

We found that BGW_1 is the best scheme of the three to use as it generates and calculates session keys much faster than DPP_1. Our scheme was immediately disqualified as a candidate because it is not secure.

In the final chapter we present ideas for future work that can be based on the work presented so far and also discuss the extent to which this thesis has completed the goals set out for it.

Chapter 6

Conclusion

6.1 Introduction

This thesis investigated the design of a bandwidth efficient and secure broadcast encryption scheme as well as a way of turning this scheme into a product entitlement system. We laid the necessary groundwork for a reader to understand the concepts required to design a secure broadcast encryption scheme. Broadcast encryption schemes from the literature were also studied and two were selected for comparison against out designed scheme. These implemented schemes were tested and evaluated. In this chapter we discuss the extent to which the original goals for the study were met and list some opportunities for future work based on the thesis.

6.2 Meeting of Goals

A set of specific goals were identified for this thesis in Section 1.2:

1. **The design of a new bandwidth efficient and secure broadcast encryption scheme**: In Chapter 3 the design of four new broadcast encryption schemes are given. These schemes are all theoretically bandwidth efficient as they require only a single group element to be broadcasted to all receivers to entitle an arbitrary set. However, these designs are not secure against collusion attacks. The bandwidth efficiency of our scheme proposed in Section 3.8 was verified with practical measurements of the size of the message header. The result of these measurements were given in Section 5.2.

2. **Develop a framework for testing and measuring performance of broadcast encryption schemes**: Chapter 4 details the software design considerations taken into account when developing a testing framework for broadcast encryption schemes. This framework was implemented to enable the completion of the final goal.

3. **The development of a framework which can be used to turn a broadcast encryption scheme into a product entitlement system**: Most of Chapter 4 is devoted to detailing the design and implementation of such a framework. This is a basic framework that is not quite ready for commercial deployment. Further work can be done on this framework to make it more user-friendly, such as providing it with a graphical user interface. In Chapter 5 this framework was also tested to verify that it works correctly.

4. **Evaluate the newly designed scheme and compare it to examples found in the literature**: The results from the testing framework were given in Chapter 5. These were the performance results of the final scheme we proposed as well as the performance results of two bandwidth efficient broadcast schemes found in the literature. We discussed these results and compared the three broadcast encryption schemes tested. An overview of the comparison is given in the next section.

6.3 Comparison to Existing Schemes

In Chapter 5 the results of a series of performance tests of our scheme and two schemes taken from the literature were given. These results were compared and discussed. For certain choices of parameters our scheme performed better than the two schemes from the literature but in many cases this advantage was lost as soon as the test parameters were increased. One major contributing factor is that our scheme derives its security from the integer factorisation problem while the schemes taken from the literature depend on the Diffie-Hellman problems on elliptic curves. This requires our scheme to use significantly larger integers than the schemes from literature for comparable security, resulting in larger network messages, greater storage requirements and longer computation times.

Message Header Size

We defined a scheme to be bandwidth efficient if the number of keys in the message header is independent of the size of the total population or entitled set size. All three schemes that were implemented are bandwidth efficient according to this definition as verified by the measurements taken. There was a slight increase in message header size when the total population is increased, but this is due the representation of the entitled set growing larger and not the addition of more keys in the header. When the absolute sizes of the message headers are compared all schemes have similar performance when the bits of security is low. Our scheme exhibits a large growth if the number of bits of security is increased while the other schemes show no growth.

Computation Time

Our scheme is the slowest of the three when creating the broadcast centre and receiver clients. In order to create a receiver client the schemes from the literature calculate much of the information once and gives it to every receiver client. In our scheme there is very little such shared information and a large set of unique information. This leads to the expected and observed large growth in computation time for our scheme when the total population is increased. Similar slowdown is observed in our scheme when the bits of security is increased for all schemes. The other schemes exhibit an increase in computation time as the population and bits of security increase, but much less so than our scheme.

All schemes require very little time when calculating the keys to be used in the message headers. When using a small number of bits of security our scheme is faster than the others but again a slowdown occurs for our scheme when increasing the bits of security. All schemes show no increase in time to calculate session keys when increasing the total population. Our scheme and BGW_1 show a similar linear increase in the time required to calculate a session key when the size of the entitled set is increased. DPP_1 shows an increase in time required as the size of the entitled set decreases.

DPP_1 shows significant increase in time required to derive a session key from the message header as the entitled set size decreases. The partial fractions based approach we introduced for use in DPP_1 significantly decreased the rate of growth when compared with the originally proposed method to derive session keys. Our scheme and BGW_1 show a slight linear increase in time required when the entitled set grows larger. When increasing the bits of security our scheme shows a significant increase in the time taken to derive a session key.

Storage requirements

All schemes require each receiver client as well as the broadcast centre to store some information about all viewers in the population. This explains the observed linear growth of the storage requirements of all schemes when increasing the total population in the system. Our integer based scheme does require significantly more storage when the number of bits of security is increased, but this can be attributed to the increase of size of the individual elements stored and not the increase in number of elements stored.

Final Recommendation

If one of the schemes tested were to be used in a product entitlement system the BGW_1 scheme, detailed in Section 3.3, would be best suited. Our proposed scheme (Section 3.8) is shown to be insecure and thus not suitable. While the BGW_1 scheme and the DPP_1

scheme (Section 3.4) have similar performance characteristics, the DPP_1 scheme requires significantly more time to derive the session key from the message header for entitled sets containing a small number of viewers.

6.4 Future Work

The study resulted in a large body of work to which there can still be made some improvements.

6.4.1 Broadcast Encryption Schemes

- It seems as if the scheme presented in Section 3.7 which sums elements from $\mathbb{Z}_N^*$ might be secure before a viewer is made part of an entitled set. If this can be proven to be secure a method may be found in which the final construction can be made secure against collusion attacks.

6.4.2 Product Entitlement Framework

- The framework for turning a broadcast encryption scheme into a product entitlement system is still basic. The cryptographic schemes used is fixed when the application is initialised. Since broadcast encryption schemes have different properties it can be beneficial to be able to change the broadcast encryption scheme used on a per broadcast basis without having to restart the application.

- The server architecture used for correctness testing over a network does not perform well when a large number of receivers are emulated. If the server architecture can be improved then more receivers can be simultaneously emulated for tests.

- The framework does not have a very user-friendly interface. This can be addressed by constructing a graphical user interface.

- Unless the framework is set up to relay and encrypt the content from a video streaming server it cannot specify a different stream to be broadcast. With the addition of a graphical user interface a possible queueing system can be implemented so that broadcasts can be taken from different files to create a program schedule easily.

- The testing framework works as required, but some minor improvements can be made. Currently only one scheme can be tested at a time and there is no queueing mechanism to run tests sequentially. As some of the tests can take several hours to complete the test might finish at a time when the testing computer is unattended. A queueing system would ensure that no time is wasted in running the tests.

- The framework allows for the specification of testing parameters on a per scheme basis, however this still requires a recompilation of the framework. This can be avoided by allowing the user to specify the testing parameters via the command line interface.

- The serialisation of the network messages before they are sent over the network can be investigated for possible improvements. Currently the product entitlement system relies heavily on Java to serialise the group elements and the containing class for the network messages. Improvements in bandwidth efficiency might be made if custom routines are created for the serialisation of the network messages that have total control over the way the messages are represented during network transport.

Bibliography

[1] Barker, E., Barker, W. and Burr, W. (2007). Recommendation for Key Management. pp. 1–142.

[2] Bell TV (2011). Bell TV Online programme guide. Online. Available at: http://tvonline.bell.ca/tvonline/servlet/CommandServlet?command=flow&processid=152#.htm

[3] Billet, O., Gilbert, H. and Ech-Chatbi, C. (2005). Cryptanalysis of a white box AES implementation. In: *Selected Areas in Cryptography*, pp. 227–240. Springer. Available at: http://www.springerlink.com/index/M6218G2DERKATWU7.pdf

[4] Boneh, D., Gentry, C. and Waters, B. (2005). Collusion resistant broadcast encryption with short ciphertexts and private keys. In: *Advances in Cryptology*, 1, pp. 258–275. Springer. Available at: http://www.springerlink.com/index/374pjlcakffdffnw.pdf

[5] Boneh, D., Goh, E. and Nissim, K. (2005). Evaluating 2-DNF formulas on ciphertexts. *Theory of Cryptography*, pp. 325–341. Available at: http://www.springerlink.com/index/wtt5caxkr94laxkg.pdf

[6] Boneh, D., Rivest, R., Shamir, A., Adleman, L. *et al.* (1999). Twenty years of attacks on the rsa cryptosystem. *Notices of the AMS*, vol. 46, no. 2, pp. 203–213.

[7] Boneh, D. and Waters, B. (2006). A Fully Collusion Resistant Broadcast , Trace , and Revoke System. In: *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 211–220. ACM. Available at: http://dl.acm.org/citation.cfm?id=1180432

[8] Burns, J. and Mitchell, C. (1994). Parameter selection for server-aided RSA computation schemes. *Computers, IEEE Transactions on*, vol. 43, no. 2, pp. 163–174. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=262121

[9] Cheung, L., Cooley, J., Khazan, R. and Newport, C. (2007). Collusion-resistant group key management using attribute-based encryption. *Group-Oriented Crypto-*

graphic Protocols, p. 23.
Available at: http://www1.hgi.rub.de/gocp07/index-Dateien/abstracts_gocp.pdf#page=23

[10] Chow, S., Eisen, P., Johnson, H. and van Oorschot, P. (2003). A white-box DES implementation for DRM applications. *Digital Rights Management*, pp. 1–15.
Available at: http://www.springerlink.com/index/V4437FAYH00KKWB8.pdf

[11] Chow, S., Eisen, P., Johnson, H. and Van Oorschot, P. (2003). White-box cryptography and an AES implementation. In: *Selected Areas in Cryptography*, pp. 250–270. Springer.
Available at: http://www.springerlink.com/index/UMTHLWNV19TDQW8V.pdf

[12] Delerablée, C., Paillier, P. and Pointcheval, D. (2007). Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In: *Pairing*, pp. 39–59. Springer.
Available at: http://www.springerlink.com/index/a253xjk457gw3611.pdf

[13] Digital Video Broadcasting Project (2010 August). DVB-S2 Fact Sheet. Online.
Available at: http://dvb.org/technology/fact_sheets/DVB-S2_Factsheet.pdf

[14] Dines, L. (1926). On positive solutions of a system of linear equations. *The Annals of Mathematics*, vol. 28, no. 1, pp. 386–392.
Available at: http://www.jstor.org/stable/10.2307/1968384

[15] Electronic Frontier Foundation (2012). Frequently Asked Questions (FAQ) About the Electronic Frontier Foundation's "DES Cracker" Machine. Online.
Available at: https://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html

[16] Fiat, A. and Naor, M. (1993 January). Broadcast Encryption. In: *Advances in Cryptology*, pp. 480–491.

[17] Goubin, L., Masereel, J. and Quisquater, M. (2007). Cryptanalysis of white box DES implementations. In: *Selected Areas in Cryptography*, pp. 278–295. Springer.
Available at: http://www.springerlink.com/index/g131074086038545.pdf

[18] Jho, N., Hwang, J., Cheon, J., Kim, M., Lee, D. and Yoo, E. (2005). One-way chain based broadcast encryption schemes. *Advances in Cryptology*, pp. 559–574.
Available at: http://www.springerlink.com/index/pp4rtulmw52gl3ex.pdf

[19] Junod, P. and Karlov, A. (2010). An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies. In: *Proceedings of the tenth annual ACM workshop on Digital rights management - DRM '10*, p. 13. ACM Press,

New York, New York, USA. ISBN 9781450300919.
Available at: http://crypto.junod.info/drm10.pdf

[20] Katz, J. and Lindell, Y. (2007). *Introduction to Modern Cryptography*. Chapman & Hall/CRC.

[21] Kirkels, B., Maas, M. and Roelse, P. (2007). A security architecture for pay-per-view business models in conditional access systems. In: *Proceedings of the 2007 ACM workshop on Digital Rights Management - DRM '07*, p. 1. ACM Press, New York, New York, USA. ISBN 9781595938848.
Available at: http://portal.acm.org/citation.cfm?doid=1314276.1314279

[22] Koblitz, N. (1987 April). Elliptic curve cryptography. *Bundesamt fur Sicherheit in der Informationstechnik*, vol. 48, no. 177, pp. 125–141.

[23] Kocher, P. and Jaffe, J. (1998). Introduction to differential power analysis and related attacks. *Lecture Notes in Computer Science*.
Available at: http://131.193.50.237/References/DPA/1998_IntroductiontoDifferentialPowerAnalysisandRelatedAttacks.pdf

[24] Matsumoto, T. and Kato, K. (1990). Speeding up secret computations with insecure auxiliary devices.
Available at: http://www.springerlink.com/index/A4D9Y3J2ML52PJVH.pdf

[25] Matt Peckham (2011 15 March). Live TV Comes to iPad, But You'll Have to Stay Home. Online.
Available at: http://techland.time.com/2011/03/15/live-tv-comes-to-ipad-but-youll-have-to-stay-home/

[26] Messerges, T., Dabbish, E. and Sloan, R. (2002). Examining smart-card security under the threat of power analysis attacks. *Computers, IEEE Transactions on*, vol. 51, no. 5, pp. 541–552.
Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1004593

[27] Miller, V.S. (2004 August). The Weil Pairing, and Its Efficient Calculation. *Journal of Cryptology*, vol. 17, no. 4, pp. 235–261. ISSN 0933-2790.
Available at: http://www.springerlink.com/index/10.1007/s00145-004-0315-8

[28] Multichoice (2011). FAQs DSTV mobile. Online.
Available at: http://www.dstvmobile.com/dstvmobile/content/en/south-africa-plus/south-africa-plus-faqs

[29] Multichoice (2012 30 March). Vital Stats. Online.
Available at: `http://www.multichoice.co.za/multichoice/view/multichoice/en/page44130`

[30] Naor, D., Naor, M. and Lotspiech, J. (2001). Revocation and Tracing Schemes for Stateless Receivers. In: *Advances in Cryptology CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 41–62. Cryptology {ePrint} Archive, Springer-Verlag, Berlin Germany.
Available at: `http://link.springer.de//link/service/series/0558/papers/2139/21390041.pdf`

[31] Nymann, J. (1975). On the probability that k positive integers are relatively prime II. *Journal of Number Theory*, vol. 7, no. 4, pp. 406–412.

[32] Pointcheval, D. (1999). New Public Key Cryptosystems based on the Dependent - RSA Problems. In: *In Eurocrypt '99, LNCS 1592*, vol. 99, pp. 239–254.

[33] PRLog (2011). Irdeto Secures Shaanxi BC&TV Transition to Digital TV with Cloaked CA. Online.
Available at: `http://www.prlog.org/11257158-irdeto-secures-shaanxi-bctv-transition-to-digital-tv-with-cloaked-ca.html`

[34] Schneier, B. (1994). Description of a new variable-length key, 64-bit block cipher (Blowfish). In: *Fast Software Encryption*, December 1993, pp. 191–204. Springer.
Available at: `http://www.springerlink.com/index/7607TU1U6R0444U9.pdf`

[35] Shannon, C.E. (1949). Communication theory of secrecy systems. *Bell System Technical Journal*, vol. 15, no. 28, pp. 656–715. ISSN 0724-6811.
Available at: `http://www.ncbi.nlm.nih.gov/pubmed/22401422`

[36] Shimbo, A. (1990). Factorisation attack on certain server-aided computation protocols for the RSA secret transformation. *Electronics Letters*, vol. 26, no. 17, pp. 16–17.
Available at: `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=83000`

[37] Smid, M. and Branstad, D. (1988 May). Data Encryption Standard: past and future. *Proceedings of the IEEE*, vol. 76, no. 5, pp. 550–559. ISSN 00189219.
Available at: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4441`

[38] Tzeng, W. and Liu, Y. (2008). Public key broadcast encryption with low number of keys and constant decryption time. *Science*, vol. 2, pp. 1–14.
Available at: `http://ir.lib.nctu.edu.tw/handle/987654321/40993`

[39] Wyseur, B. (2008). *White-Box Cryptography*. Ph.D. thesis.
Available at: `http://www.cosic.esat.kuleuven.be/publications/talk-98.pdf`

[40] Wyseur, B., Michiels, W., Gorissen, P. and Preneel, B. (2007). Cryptanalysis of white-box DES implementations with arbitrary external encodings. In: *Proceedings of the 14th international conference on Selected areas in cryptography*, pp. 264–277. Springer-Verlag. ISBN 3540773592.
Available at: `http://portal.acm.org/citation.cfm?id=1784881.1784898`

[41] Zhou, Z. and Huang, D. (2010). On efficient ciphertext-policy attribute based encryption and broadcast encryption. *Proceedings of the 17th ACM conference on*, pp. 1–19.
Available at: `http://dl.acm.org/citation.cfm?id=1866420`

[42] Zou, X., Dai, Y. and Bertino, E. (2008). A practical and flexible key management mechanism for trusted collaborative computing. In: *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 538–546. IEEE.